

Designing new models and algorithms to improve order picking operations

Vom Fachbereich Rechts- und Wirtschaftswissenschaften der
Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doctor rerum politicarum (Dr. rer. pol.)
genehmigte

Dissertation

vorgelegt von
Ivan Žulj, M. Sc.

Datum der Einreichung: 30.03.2020

Erstgutachter: Univ.-Prof. Dr. Michael Schneider
Zweitgutachter: Univ.-Prof. Dr. Christoph Glock
Drittgutachterin: Univ.-Prof. Dr. Katja Schimmelpfeng

Darmstadt, 2020

Žulj, Ivan: Designing new models and algorithms to improve order picking operations

Darmstadt, Technische Universität Darmstadt,

Jahr der Veröffentlichung der Dissertation auf TUpriints: 2020

Tag der mündlichen Prüfung: 28.05.2020

Veröffentlicht unter CC BY-SA 4.0 International

<https://creativecommons.org/licenses/>

Contents

Contents	I
Acknowledgments	IV
List of figures	VI
List of tables	VIII
List of abbreviations	X
1 Overview of the thesis	1
1.1 Current challenges in order picking	1
1.2 Contributions to research and practice	3
1.3 Organization	9
2 Fundamentals	11
2.1 Warehouse operations	12
2.2 Classification of order picking systems	13
2.3 Central components of picker-to-parts systems	15
2.3.1 Warehouse layout	15
2.3.2 Order picking process	17
2.4 Common planning objectives in order picking	19
2.5 Planning problems in order picking	21
2.5.1 Warehouse layout design	21
2.5.2 Item storage assignment	22
2.5.3 Picker routing	25
2.5.3.1 The standard picker routing problem	25
2.5.3.2 Variants of picker routing problems	30

2.5.4	Order batching	34
2.5.4.1	Introduction	34
2.5.4.2	The standard order batching problem	36
2.5.4.3	Variants of order batching problems	43
2.5.5	Zone picking	49
2.5.6	AGV-assisted order picking	51
2.6	Metaheuristics used in this dissertation	53
2.6.1	Classification of metaheuristics	53
2.6.2	Tabu search	54
2.6.3	Large neighborhood search	56
2.6.4	Adaptive large neighborhood search	58
3	A metaheuristic hybrid of adaptive large neighborhood search and tabu search for the standard order batching problem	61
3.1	Problem description	62
3.2	Solution method	63
3.2.1	Initial solution	63
3.2.2	Adaptive large neighborhood search component	64
3.2.3	Tabu search component	68
3.3	Computational studies	68
3.3.1	Parameter setting	68
3.3.2	Benchmark instances	69
3.3.3	Scaling behavior of the metaheuristic hybrid	71
3.3.4	Effect of algorithmic components	72
3.3.5	Comparison to the state-of-the-art	74
3.4	Summary and conclusion	83
4	Picker routing and storage assignment strategies for precedence-constrained order picking	84
4.1	Problem description	85
4.2	Solution algorithm	86
4.3	Practical case study	88
4.3.1	Case description	89
4.3.2	Results of the case study	91
4.4	Influence of different problem parameters	97
4.4.1	Test instances	98

4.4.2	Results of the numerical study	98
4.5	Summary and conclusion	103
5	Order batching and batch sequencing in an AGV-assisted picker-to-parts system	105
5.1	Problem description	106
5.2	Solution approach	119
5.2.1	Metrics used in the solution approach	119
5.2.2	Adaptive large neighborhood search for batching customer orders	120
5.2.2.1	Initial solution	120
5.2.2.2	Adaptive large neighborhood search component	122
5.2.3	Construction heuristic for deciding the batch processing sequence	125
5.3	Numerical studies	126
5.3.1	Instance generation	126
5.3.2	Parameter setting for the adaptive large neighborhood search component	128
5.3.3	Effect of the simulated annealing-based acceptance criterion	129
5.3.4	Comparison of the solution method to a commercial solver	130
5.3.5	Experiments on the AGV fleet size and speed	131
5.4	Summary and conclusion	136
6	Summary, conclusion, and future research	140
6.1	Summary and conclusion	140
6.2	Outlook on future research	144
	Bibliography	i
	Appendices	xi
A	Alternative mathematical model for the AGV-assisted order picking problem	xii
B	Detailed computational results of the comparison methods on small instances of our AGV-assisted order picking problem	xvi

Acknowledgments

This dissertation was written during my work as a research assistant at the Department of Logistics Planning and Information Systems at the Technical University of Darmstadt and at the Department of Procurement and Production at the University of Hohenheim. While writing this dissertation, I received tremendous support from a lot of different people, so now it is time to acknowledge their contribution.

First and foremost, I am grateful to my PhD supervisor Prof. Dr. Michael Schneider. Although Michael would probably say that I should rather spend my time on the next research project, I want to take this opportunity to express my deepest gratitude. Despite his busy schedule, Michael has always found time for excellent suggestions, fruitful discussions, constructive comments, and valid questions. He encouraged me to aim for better results in his direct and entertaining manner (“Stop complaining, and let’s get back to work!”) and provided exemplary guidance throughout the entire dissertation. I want to thank him for sharing his great knowledge and experience with me as well as for enabling interesting collaborations with leading industrial companies. Even though this PhD thesis is almost finished, I hope to continue to learn from him both academically and personally.

Furthermore, I would like to express my gratitude to Prof. Dr. Christoph Glock for acting as second reviewer of my dissertation. I am very thankful for all the valuable advice I received especially during the joint research project on precedence-constrained order picking and also for his assistance in organizing the dissertation’s defense.

I owe many thanks to Prof. Dr. Katja Schimmelpfeng: for writing the third review for my dissertation, for the exceptional support, for the steady encouragement, for always giving me the necessary freedom to work on my research projects, and for the excellent working atmosphere at her department.

I also thank the members of my PhD committee for generously offering their time.

I would like to deeply thank Dr. Dominik Goeke, Dr. Eric Grosse, Sergej Kramer, and Dr. Hagen Salewski, who coauthored the publication of selected parts of this dissertation in scientific journals.

Likewise, I am grateful to my (former) colleagues Hendrik Butemann, Gerriet Fuchs, Christopher Haager, and Rebecca Hummel, who always supported me and also contributed to a great working atmosphere. I particularly thank Hendrik Butemann for the fruitful scientific and non-scientific conversations. Although the latter certainly led to some delay in the submission of this dissertation, I do not regret it at all. His help and motivation directed my research towards promising areas and helped me to avoid dead ends. I also thank Gerriet Fuchs and Christopher Haager for their critical questions, valuable input, and thoughts on my dissertation. Moreover, I would like to thank Rebecca Hummel for making a substantial contribution to this dissertation. I appreciate her selflessness and wish her all the best for the completion of her dissertation.

Finally, I am very grateful to my friends and my family for the unconditional support and encouragement throughout the last years.

Ivan Žulj, Leinfelden-Echterdingen, March 2020

List of figures

1.1	An example of a rectangular single-block warehouse layout.	4
2.1	Overview of typical warehouse areas.	12
2.2	A classification of order picking systems.	14
2.3	An example of an order picking process in a picker-to-parts system. .	18
2.4	Typical distribution of order picking time.	20
2.5	Overview of common item storage assignment strategies.	22
2.6	Example of the resulting order picking tours for different picker routing strategies.	28
2.7	An example for reducing the total tour length by order batching assuming the S-shape picker routing strategy in a single-block parallel-aisle warehouse.	35
2.8	SSI Schäfer's FTS 2PICK™.	52
2.9	TS in pseudocode.	54
2.10	LNS in pseudocode.	57
2.11	ALNS in pseudocode.	59
3.1	Overview of the ALNS×TS algorithm.	63
4.1	Order picking tours for different end locations of a heavy subtour and start locations of a light subtour, respectively.	88
4.2	Weight-based storage assignment strategies.	94
4.3	Performance of H-PRSW/O, E-PRSW/O, and E-PRSW for different storage assignments and sorting effort scenarios.	96
5.1	Warehouse layout assumed for our AOPP.	107
5.2	Order picking process from the perspective of an order picker in the AGV-assisted order picking system.	113

5.3	Order picking process from the perspective of an AGV in the AGV-assisted order picking system.	114
5.4	An example illustrating precedence relationships between two order pickers and a single AGV when processing a single batch.	115
5.5	Overview of the ALNS algorithm.	121
5.6	Overview of the NEH-based algorithm.	126
5.7	Average total tardiness that results on the instances assuming a warehouse with 10 picking aisles.	136
5.8	Average total tardiness that results on the instances assuming a warehouse with 15 picking aisles.	138
5.9	Average total tardiness that results on the instances assuming a warehouse with 20 picking aisles.	138

List of tables

2.1	Overview of single picker routing problems.	33
2.2	Exact and metaheuristic solution approaches for the standard order batching problem.	42
2.3	Overview of order batching problems integrating the sequencing of batches (BS), the assignment of batches to order pickers (BA), and/or picker routing (PR).	47
2.4	Overview of solution approaches for order batching problems integrating the sequencing of batches, the assignment of batches to order pickers, and/or a picker routing problem.	48
3.1	Comparison of ALNS×TS results using C&W(i) and C&W(ii) on the benchmark set UDD/S-shape.	65
3.2	Overview of the parameter setting of ALNS×TS used in the computational studies.	70
3.3	Results of different ALNS×TS configurations.	73
3.4	Performance of ALNS and TS as standalone methods in comparison to ALNS×TS on the benchmark set UDD/S-shape.	74
3.5	Performance of ALNS×TS on the benchmark set UDD/S-shape. . . .	76
3.6	Performance of ALNS×TS on the benchmark set CBD/S-shape. . . .	78
3.7	Performance of ALNS×TS on the benchmark sets UDD/largest gap and CBD/largest gap.	79
3.8	Performance of ALNS×TS in comparison to ILST on the benchmark set UDD/S-shape.	81
3.9	Performance of ALNS×TS in comparison to ILST on the benchmark set CBD/S-shape.	82
4.1	Performance of the picker routing strategies assuming random storage.	92
4.2	Performance of the picker routing strategies for different W-SASs. . .	95

4.3	Performance of the strategies for different problem parameters in a warehouse with 10 picking aisles.	99
4.4	Performance of the strategies for different problem parameters in a warehouse with 25 picking aisles.	100
4.5	Performance of the strategies for different problem parameters in a warehouse with 50 picking aisles.	101
5.1	Overview of the notation used in the AOPP-BE model.	116
5.2	Overview of the parameter setting of the ALNS component used to generate a batching solution.	129
5.3	Effect of using an SA-based acceptance criterion after the ALNS phase on the large-sized instances in set L	130
5.4	Performance of ALNS/NEH in comparison to CPLEX on small-sized instances in set S	132
5.5	Change in objective function values (in percent).	137
A.1	Overview of the notation used in the AOPP-BI model.	xiii
B.1	Comparison of the results obtained with CPLEX and ALNS/NEH on the small-sized instances assuming five customer orders.	xvii
B.2	Comparison of the results obtained with CPLEX and ALNS/NEH on the small-sized instances assuming six customer orders.	xviii
B.3	Comparison of the results obtained with CPLEX and ALNS/NEH on the small-sized instances assuming seven customer orders.	xix

List of abbreviations

ABHC	attribute-based hill climber
AGV	automated guided vehicle
ALNS	adaptive large neighborhood search
AOPP	AGV-assisted order picking problem
AOPP-BE	AOPP in which the feasible batches not exceeding the capacity of the picking device are generated in advance
AOPP-BI	AOPP in which feasible batches are considered implicitly when solving a test instance
AP	aspiration plus
BI	best improvement
BKS	best known solution
CBD	class-based demand
CPLEX	IBM ILOG CPLEX Optimizer
C&W	Clarke and Wright
E-PRS	exact picker routing strategy
E-PRSW	exact picker routing strategy, which considers the precedence constraint
E-PRSW/O	exact picker routing strategy, which does not consider the precedence constraint
FCFS	first-come first-served
GA	genetic algorithm
GGA	group-oriented genetic algorithm
H-PRS	heuristic picker routing strategy
H-PRSW/O	heuristic picker routing strategy, which does not consider the precedence constraint
IGA	item-oriented genetic algorithm
ILS	iterated local search
ILST	iterated local search with tabu thresholding

LNS	large neighborhood search
LS	local search
LU	length unit
NEH	Nawaz, Ensore, and Ham
OBP	order batching problem
SA	simulated annealing
SE	sorting effort
TS	tabu search
UDD	uniformly distributed demand
WMS	warehouse management system
W-SAS	weight-based storage assignment strategy

Chapter 1

Overview of the thesis

1.1 Current challenges in order picking

Order picking describes the process of retrieving inventory items from their storage locations to satisfy customer orders (de Koster et al. 2007). It has long been identified as the most laborious and costly warehouse operation, accounting for up to 65% of total warehouse operating costs (Petersen and Schmenner 1999, Coyle et al. 2002). Order picking is a crucial factor for the competitiveness of a supply chain because inadequate order picking performance (e.g., long delivery times) results in customer dissatisfaction and high costs (e.g., labor cost) for the warehouse (Wäscher 2004).

It is estimated that about 80% of all order picking systems in Western Europe follow the traditional picker-to-parts setup, in which order pickers walk (or travel) through the warehouse to retrieve the requested items from their storage locations (de Koster et al. 2007, Napolitano 2012). While the investment costs for such systems are rather low, the major drawback is the large fraction of unproductive picker walking time with 50% and more of total order picking time (de Koster et al. 2007, Tompkins et al. 2010). Although technologies to automate order picking exist (see, e.g., Azadeh et al. 2019), warehouse managers rely on human order pickers because of their inherent flexibility and ability to adapt to changes in real-time in contrast to automated systems (Grosse et al. 2014).

The steadily increasing global retail sales volumes (Statista 2019), however, force warehouse managers to improve the performance of their warehouse operations to fulfill customer requirements (e.g., responsive order fulfillment) and to gain advan-

tages over competitors. Here, warehouse managers often face the following challenges:

- Tight delivery schedules because of (contractually agreed or promised) next or even same-day deliveries put increasing burden on warehouse operations and lead to highly time-critical order fulfillment processes (Boysen et al. 2019a).
- E-commerce warehouses receive a large number of customer orders, which only consist of a few items each (Boysen et al. 2019b). Traditional picker-to-parts systems are rarely suitable for these requirements. For example, in picker-to-parts systems in which the pick-by-order strategy is applied, customer orders are picked individually on a single order picking tour starting from the depot, proceeding along the storage locations defined by the respective customer order, and ending at the depot. Due to the low pick density per order picking tour in such scenarios, the part of unproductive work an order picker spends on each customer order while walking (or traveling) through the warehouse is often large. The resulting loss of throughput makes it difficult to meet the customers' expectations for fast delivery (Boysen et al. 2019a).
- Contrary, in specific branches (e.g., in the online grocery sector), customer orders are likely to consist of dozens of items (Valle et al. 2017, Boysen et al. 2019a). Handling large customer orders within tight delivery schedules is difficult for warehouse managers because such customer orders require longer picking times compared to small customer orders.

Research papers on order picking strategies and algorithms have mostly concentrated on traditional picker-to-parts systems (Chabot et al. 2017). Likewise, constraints arising in real-world applications have often been neglected. In recent years, research has started to consider more realistic characteristics of real-world warehouse operations like item-specific characteristics, such as fragility or weight, and also human factors, such as physical workload (see, e.g., Chackelson et al. 2013, Grosse et al. 2015, Chabot et al. 2017, Matusiak et al. 2017, Glock et al. 2019). A frequently encountered constraint in real-world applications of order picking concerns precedence constraints (Matusiak et al. 2014). Such constraints define that certain items need to be collected before other items, e.g., non-food items have to be placed underneath food items on a pallet to avoid contamination. Obviously, any of these practical restrictions further complicates the planning of warehouse operations.

This dissertation aims at designing new models and algorithms to improve order picking efficiency and to support managerial decisions on facing the warehousing challenges described above. We discuss this in detail in the next section, which is devoted to our contributions to research and practice.

1.2 Contributions to research and practice

The contributions of this thesis are detailed in the following.

Order batching

To reduce unproductive picker walking and to meet tight delivery schedules, the pick-by-batch strategy can be applied. Instead of picking customer orders individually on different order picking tours, customer orders are grouped into a batch of customer orders jointly picked on a single tour. Effectively batching customer orders leads to a higher pick density per tour compared to the pick-by-order strategy and thus increases order picking performance.

In this context, we study the so-called standard order batching problem (standard OBP), which can be defined as follows: The standard OBP considers a picker-to-parts system in a rectangular single-block warehouse (see Figure 1.1), from which the requested items of a given set of customer orders have to be retrieved. In the warehouse, parallel picking aisles of equal length and width are connected by a cross aisle at the front and at the rear of the picking aisles. Items are stored in storage locations arranged along both sides of the picking aisles, and each item is available from exactly one storage location. In the standard OBP, an order picker can retrieve items from both sides of a picking aisle without performing additional movements. The depot may be located anywhere along the front or the rear cross aisle. Figure 1.1 depicts an example of a rectangular single-block warehouse, in which 120 different items are stored in six picking aisles. Each picking aisle contains 20 storage locations, 10 on the left and 10 on the right of the picking aisle. The depot is located below the entry of the leftmost picking aisle in the front cross aisle.

The standard OBP assumes that customer orders may be combined into batches until the capacity of a picking device (e.g., a picking cart) is exhausted. The picking device is used to transport the retrieved items of a batch through the warehouse. In addition, the items of a customer order cannot be distributed among different batches because splitting may result in unacceptable sorting effort. Order picking

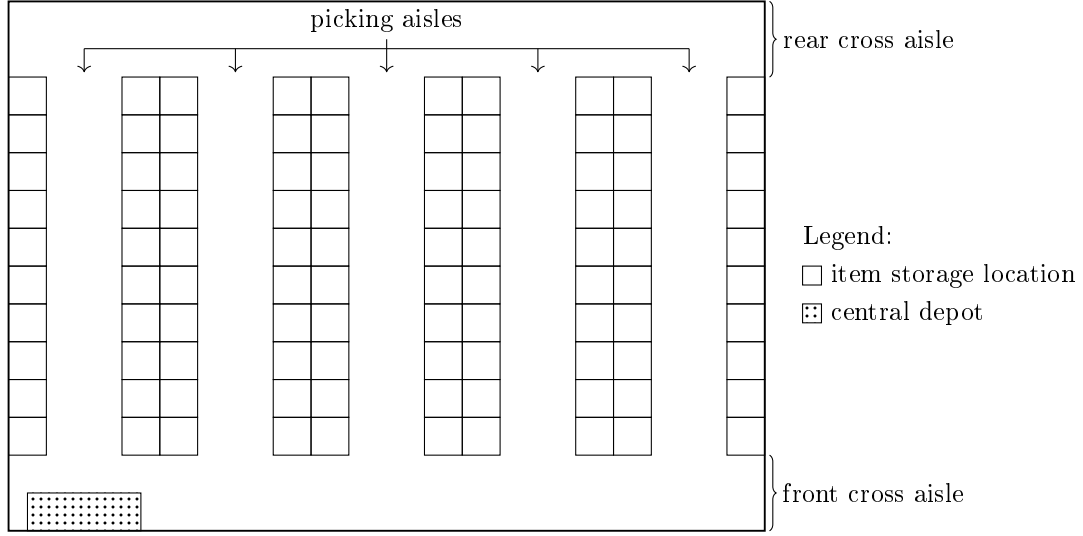


Figure 1.1: An example of a rectangular single-block warehouse layout.

tours start from the depot, proceed along the storage locations defined by the respective batch, and end at the depot. A given routing algorithm determines an order picker's tour through the warehouse and the retrieval sequence of the items from their storage locations (see Section 2.5.3).

The goal of the standard OBP is to group customer orders into batches such that the total length of all order picking tours for collecting the items of the batches from their storage locations is minimized. The standard OBP is NP-hard if the number of orders per batch is greater than two, and the exact solution methods proposed in the literature are not able to consistently solve larger instances.

With respect to the standard OBP, the main contributions of this thesis are as follows:

- First, to solve larger OBP instances within short runtime, we develop a hybrid of adaptive large neighborhood search (ALNS) and tabu search (TS), denoted as ALNS×TS. An advantage of the hybridization is that it combines the diversification capabilities of ALNS and the intensification capabilities of TS. We assess the performance of the hybridization in numerical studies and show that ALNS×TS clearly outperforms ALNS and TS as standalone methods. To the best of our knowledge, our method is the first ALNS and the first hybrid meta-heuristic designed for the standard OBP. The relative simplicity and inherent flexibility renders our approach interesting for warehouse operators.

- Second, we investigate the performance of ALNS×TS on the standard OBP benchmark sets available in the literature and on newly generated large-scale instances with up to 600 customer orders. We make the complete set of benchmark instances available for download, we explain how these instances were interpreted by previous authors to make results comparable, and we report detailed results of our computational experiments to give comparison values for future methods investigating the standard OBP. As described in Section 3.3, this has not been done by previous authors.
- Third, we compare the performance of ALNS×TS to all previously published methods that have been tested on (any subset of) the standard OBP instances from the literature. Our ALNS×TS is able to outperform all these methods with respect to the average solution quality and runtime over all benchmark sets. For the more practically relevant instances with a larger number of customer orders and larger capacities of the picking device, ALNS×TS shows the clearest advantages compared to the existing methods. Furthermore, ALNS×TS is able to solve newly generated large-scale instances with reasonable runtimes and convincing scaling behavior and robustness.

Precedence-constrained order picking

The next problem examined in this thesis is inspired by a practical case observed in a rectangular single-block warehouse of a German manufacturer of household products. Here, the items to be picked can be roughly distinguished into light (fragile) and heavy (robust) items. The case company applies a random storage assignment strategy according to which items are randomly assigned to storage locations of the warehouse.

To prevent damage to light items, order pickers are not permitted to put heavy items on top of light items. Currently, an order picking tour is determined by applying a heuristic picker routing strategy (H-PRS) that does not consider this precedence constraint. As a result, an order picker collects the items of a customer order into a plastic box without stacking the items on top of each other. After having retrieved the requested items of a customer order from the shelves of the warehouse, the order picker travels back to the central depot, where she¹ packs the

¹For the sake of readability, we have decided to speak exclusively of female order pickers. Of course, an order picker can be of any other gender.

collected items into a cardboard box that is used for shipping the items such that the precedence constraint is respected. We refer to this picker routing strategy as $H\text{-}PR_{S_{W/O}}$, where W/O indicates that the precedence constraint is not considered during the collection of the requested items.

As our literature review on picker routing problems (PRPs) shows, works considering precedence constraints in picker routing are rare. Likewise, the impact of storage assignment on the performance of a picker routing strategy is hardly investigated although the assignment of items to storage locations influences the tour length of an order picker for completing a customer order as well. Often, the performance of a picker routing strategy is discussed by assuming random storage.

With respect to precedence-constrained order picking, we make the following contributions:

- To avoid that items have to be sorted after the retrieval process, we propose a picker routing strategy that incorporates the described precedence constraint and collects heavy items before light items. To shorten travel distances in the warehouse, we determine an optimal order picking tour, which leads to the minimum tour length for collecting heavy items before light items on a single order picking tour. In the following, $E\text{-}PR_{S_W}$ denotes our exact picker routing strategy, where W indicates that the precedence constraint is considered during the collection of the requested items. Furthermore, we suggest different weight-based storage assignment strategies and investigate their impact on the performance of the proposed picker routing strategy.
- The performance of $E\text{-}PR_{S_W}$ is assessed on a dataset provided to us by the case company. We compare $E\text{-}PR_{S_W}$ to (i) $H\text{-}PR_{S_{W/O}}$, i.e., the strategy used in the warehouse of the manufacturer, and to (ii) an exact picker routing approach that neglects the given precedence constraint (called $E\text{-}PR_{S_{W/O}}$).

The results of the analysis show that the current order picking process of the case company can be improved in the following aspects: First, $E\text{-}PR_{S_W}$ enables the order pickers to place the retrieved items directly in the cardboard boxes required for shipping the items and thus avoids the use of plastic boxes and the sorting of items upon return to the depot. Second, we reduce the average travel tour length of an order picker compared to the picker routing strategy of the case company. Third, we find that the storage assignment strategy significantly affects the performance of $E\text{-}PR_{S_W}$. By separating heavy items and light items

in the warehouse and allocating heavy items to storage locations that are arranged close to the depot, a strong reduction of the average tour length can be achieved.

- We generate new problem instances to investigate the influence of different problem parameters (warehouse size, share of heavy and light items per customer order, and number of requested items per customer order) on the performance of $H\text{-}PR_{S_{W/O}}$, $E\text{-}PR_{S_{W/O}}$, and $E\text{-}PR_S$.

The results of the numerical studies show that $E\text{-}PR_S$ provides a convincing performance even if we compare our strategy to the exact solution approach that neglects the precedence constraint ($E\text{-}PR_{S_{W/O}}$). Moreover, $E\text{-}PR_S$ shows the most robust solution quality for the problem instances with different characteristics.

- Despite the complexity of implementing precedence constraints in order picking in general, $E\text{-}PR_S$ is easy to understand for order pickers as it follows a straightforward and non-confusing routing scheme and thus reduces the potential for errors in order picking.

AGV-assisted order picking

The last problem addressed in this thesis is inspired by a warehouse of a German automotive original equipment manufacturer with a traditional picker-to-parts setup. A set of customer orders is given, each associated with a due date until which the items of the customer order are to be collected. Because each customer order consists of only a few items, a batching of customer orders is applied to increase order picking efficiency. Moreover, to avoid unnecessary trips from the picking area of the warehouse back to the central depot, order pickers are supported by a fleet of automated guided vehicles (AGVs), where an AGV transports the items of a single batch from the picking area to the depot.

The warehouse is partitioned into disjoint zones, each with one order picker assigned to it. An order picker collects those items of a batch that are stored in her zone and transports them to a handover location, where she passes the items to an AGV. Note that zone picking can speed up order picking because an order picker does not travel between the zones, i.e., each order picker only traverses smaller areas of the warehouse. In parallel, an AGV is equipped with bins at the central depot, where each bin is associated with the items of a single customer order to avoid order

consolidation after picking. An AGV autonomously drives to a handover location, parks, displays its demand, and waits until the order picker passes the items and confirms the pick. Then, the order picker returns to her zone to retrieve the items of the next batch, and the AGV drives to other handover locations or it returns to the central depot if all items of the batch have been picked. Here, the shipping of the customer orders is prepared, and the AGV is again equipped with empty bins so that it can process the next batch.

Our AGV-assisted order picking problem (AOPP) decides on the grouping of customer orders into batches, the sequence according to which batches should be processed, and the assignment of batches to AGVs such that the total tardiness of all customer orders (i.e., the extent to which the due dates are violated) is minimized. To the best of our knowledge, the AOPP constitutes a novel setting, which has not been explored in the literature so far but is highly relevant in practice. Our contributions can be summarized as follows:

- We propose two mixed integer programming formulations for the AOPP: In the first modeling approach (referred to as AOPP-BE model), all batches not exceeding the picking device capacity (called feasible batches) are generated explicitly before solving a specific test instance. In the second model (referred to as AOPP-BI model), feasible batches are considered implicitly when solving a test instance.
- Because our problem setting extends the standard OBP, this problem is NP-hard. We focus on the development of an effective and efficient solution approach to provide solutions for large problem instances. To this end, we propose a two-stage heuristic consisting of an ALNS component for batching customer orders and an adaption of the well-known Nawaz, Enscore, and Ham heuristic (called NEH heuristic) for sequencing the batches. We denote our solution approach as ALNS/NEH.
- On newly generated large-sized instances, we analyze the impact of using a simulated annealing-based (SA-based) acceptance criterion after the ALNS phase instead of simply accepting improving solutions. The studies performed on these instances show the positive impact of using the SA-based acceptance criterion on the solution quality.
- We design a set of small instances to compare the performance of ALNS/NEH to those of the optimization software IBM ILOG CPLEX Optimizer (CPLEX).

The results clearly demonstrate the ability of ALNS/NEH to find high-quality solutions within a fraction of a second.

- To provide managerial insights, we conduct several computational experiments examining the effect of increasing the AGV fleet size and varying traveling and walking speed ratios between AGVs and order pickers on the objective of minimizing the total tardiness of all customer orders. The experiments indicate that a slight increase in the speed ratio or the fleet size results in large improvements of the total tardiness.

1.3 Organization

The remainder of this dissertation is structured as follows. In Chapter 2, the conceptual and methodological fundamentals that are relevant for the problems addressed in this work are introduced. First, we give a description of warehouse operations with a focus on order picking. Second, we present a classification of order picking systems which differentiates these systems according to whether mainly human order pickers or automated machines are involved in the order picking. Because order picking systems which mainly employ human order pickers lie at the core of all problems addressed in this work, we describe their central components. Subsequently, we present frequently used planning objectives in order picking. Next, we detail the central planning problems in picker-to-parts systems, and we review the related literature on these problems. Last, the metaheuristic paradigms used in this dissertation are introduced.

Chapter 3 provides a mathematical model formulation of the standard OBP and describes our ALNS \times TS to solve the problem. To investigate the performance of our metaheuristic hybrid, we generate large-scale instances and perform extensive numerical studies on these instances and on the standard OBP benchmark sets available in the literature. The effect of the algorithmic components is assessed by comparing the performance of ALNS and TS as standalone methods with those of the hybridization of these components on a benchmark set from the literature. Moreover, we conduct an extensive performance comparison of ALNS \times TS to all previously published methods that have been tested on (any subset of) the standard OBP instances from the literature.

In Chapter 4, we first detail the practical case, which is characterized by a precedence constraint in the order picking. Then, we introduce our exact solution

algorithm to evaluate E-PRSW. Next, we present our experimental results, which are obtained on a dataset provided to us by the case company. Moreover, we examine the impact of different item storage assignment strategies as well as different problem parameters on the performance of the picker routing strategies (H-PRSW/O, E-PRSW/O, and E-PRSW) on newly designed instances. Last, we derive managerial insights for precedence-constrained order picking.

Chapter 5 presents a mixed integer programming formulation for the AOPP and details our ALNS/NEH. In extensive computational experiments, (i) we analyze the influence of the SA-based acceptance criterion used by ALNS/NEH on the solution quality, (ii) we assess the performance of ALNS/NEH compared to CPLEX, and (ii) we give managerial insights with respect to AGV-assisted order picking.

Finally, we summarize the findings of this dissertation and give an outlook on future research opportunities in Chapter 6.

Chapter 2

Fundamentals

Some of the contents of this chapter are included in similar form in the following publications:

- I. Žulj, S. Kramer, and M. Schneider. A hybrid of adaptive large neighborhood search and tabu search for the order batching problem. *European Journal of Operational Research*, 264(2):653–664, 2018.
- I. Žulj, C. H. Glock, E. H. Grosse, and M. Schneider. Picker routing and storage assignment strategies for precedence-constrained order picking. *Computers & Industrial Engineering*, 123:338–347, 2018.

This chapter is devoted to the conceptual and methodological fundamentals that are relevant for the problems addressed in this dissertation. In Section 2.1, we give an overview of warehouse operations with a focus on order picking. Section 2.2 classifies the different order picking systems according to whether human order pickers or automated machines are mainly involved in the order picking. Because we focus on picker-to-parts systems, which are prevalent in the literature and in practice, Section 2.3 describes their central components. Section 2.4 presents frequently used planning objectives in order picking, and Section 2.5 discusses the central planning problems in picker-to-parts systems. To this end, we focus on warehouse layout design, storage assignment methods, picker routing strategies, order batching, zoning, and AGV-assisted order picking. Moreover, we review the scientific literature on these problems. Finally, Section 2.6 provides an overview of the metaheuristic solution methods applied in this dissertation, namely TS and ALNS.

2.1 Warehouse operations

Warehouse operations comprise receiving, storing, picking, sorting, packing, and shipping items (de Koster et al. 2007, Tompkins et al. 2010). Usually, a warehouse is divided into areas in which these operations are performed. Figure 2.1 illustrates a schematic representation of a warehouse containing a receiving, a reserve storage, a forward storage, a sorting and packing, and a shipping area. The areas are symbolized by rectangles, and item flows through the warehouse are indicated by arrows.

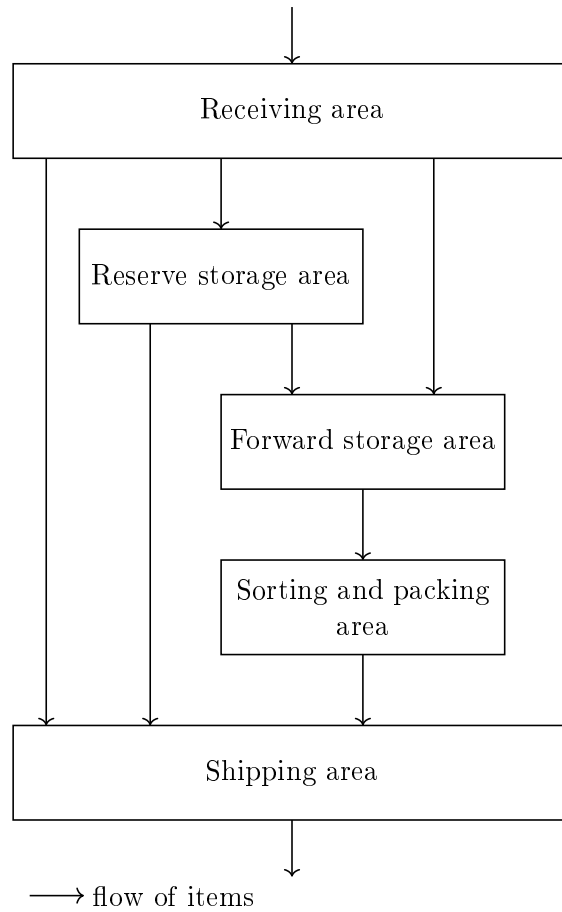


Figure 2.1: Overview of typical warehouse areas (based on Tompkins et al. (2010) and Koch (2014)).

In the following, we briefly describe these areas and the item flows through the warehouse. Activities in the receiving area include the unloading of the received items from transport carriers, quantity and quality control, updating the inventory

record, repackaging (e.g., of items received on pallets into storage modules used in the warehouse), and the transfer either to the storage or directly to the shipping area via cross docking (de Koster et al. 2007).

The storage area of a warehouse is typically divided into a reserve (or bulk) and a forward storage area (Strack and Pochet 2010). The reserve storage area stores items in bulk often in far-distant pallet racks (Walter et al. 2013). It is used for picking items which are requested in large quantities and which are not assigned to the forward storage area. The bulk stock stored in the reserve storage area also serves for replenishing the forward storage area in the case of a stock-out (van den Berg et al. 1998). The forward storage area stores small quantities of items in a compact area in easily accessible storage modules (e.g., gravity flow racks). In contrast to the reserve storage area, the forward storage area is used to pick small quantities of highly demanded items (Gu et al. 2007, Strack and Pochet 2010, Park 2012). We refer the reader to Hackman et al. (1990), van den Berg et al. (1998), and Walter et al. (2013) for details on assigning items to the reserve and/or forward storage area.

Order picking is the main operation in most warehouses (de Koster 2015). It involves the process of retrieving items from their storage locations within the reserve and/or forward storage area in response to a specific customer request. Because bulk retrievals from the reserve storage area are not considered in this dissertation, order picking relates to removals from the forward storage area, called picking area in the following. In order picking, sorting of the items can be necessary, i.e., if several customer orders are picked together, the items have to be sorted according to the respective customer orders. Sorting also includes the consolidation of items, for example, if a customer order is split and its items are picked by multiple order pickers. A detailed description of an order picking process is given in Section 2.3.2. Sorting and consolidation are described in Section 2.5.4.

Before the customer orders can be shipped to the customers, they are checked (e.g., for completeness), packed, and finally loaded onto transport carriers at the shipping area (Rouwenhorst et al. 2000, Park 2012).

2.2 Classification of order picking systems

In the literature, various alternatives can be found to classify the wide range of order picking systems (see, e.g., de Koster et al. 2007, Dallari et al. 2009, Gudehus 2012).

Figure 2.2 shows a common classification which differentiates order picking systems according to whether mainly human operators or automated machines are involved in the order picking (de Koster et al. 2007).

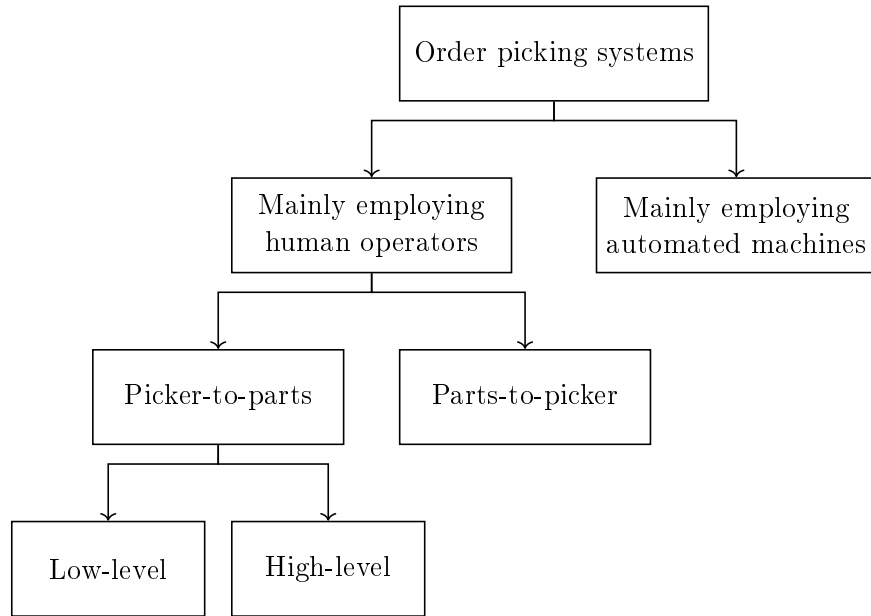


Figure 2.2: A classification of order picking systems (based on de Koster et al. (2007)).

Order picking systems which mainly employ human operators can be distinguished into picker-to-parts systems and parts-to-picker systems. In picker-to-parts systems, order pickers walk or ride through the warehouse to one or more storage locations to retrieve the items requested by customers. These systems can be further divided into low-level picker-to-parts systems and high-level picker-to-parts systems. Low-level picker-to-parts systems store items in low-level racks, in bins, or on pallets. High-level picker-to-parts systems employ high storage racks to store items. Contrary to low-level picker-to-parts systems, in which items are directly accessible for an order picker, in high-level picker-to-parts systems, an order picker is often moved to a storage location by a vehicle with a lifting platform. Throughout the dissertation, we concentrate on analyzing low-level picker-to-parts systems because they are prevalent in practice (see, e.g., de Koster et al. 2007).

Parts-to-picker systems include automated storage and retrieval systems, in which items are delivered by, e.g., aisle-bound cranes to stationary order pickers (Wäscher 2004, de Koster et al. 2007). Usually, items are provided in unit loads (e.g., pallets

or bins), order pickers remove the requested items, and the aisle-bound crane returns the remaining items to their storage location in the warehouse.

Although there are technologies to automate order picking, order picking systems which mainly employ automated machines are rarely found in practice (Wäscher 2004). These systems are used in the case of valuable, small, and sensitive items (de Koster et al. 2007).

2.3 Central components of picker-to-parts systems

2.3.1 Warehouse layout

Rectangular warehouse layouts with parallel picking aisles are common both in the literature and in practice (see, e.g., Ratliff and Rosenthal 1983, Bozer and Kile 2008, Henn and Wäscher 2012, Goeke and Schneider 2018). In a rectangular single-block warehouse, parallel picking aisles are connected by an orthogonal cross aisle at the front and at the rear of the picking aisles (see Section 1.2, Figure 1.1). The part of the picking area which is enclosed by these two cross aisles forms a so-called block. Items are stored in storage locations arranged on the left and on the right of each picking aisle. Cross aisles do not contain storage locations, but they allow order pickers to move from one picking aisle to another to retrieve the requested items. Warehouse layouts with these characteristics and with picking aisles of equal length and width as well as storage locations of identical size are referred to as single-block parallel-aisle warehouses in the following.

Single-block parallel-aisle warehouse layouts can be extended by additional cross aisles to form two-block, three-block, or, generally, multi-block parallel-aisle warehouse layouts, in which v cross aisles constitute $v-1$ blocks. Single-block and multi-block parallel-aisle warehouses are often called conventional warehouses in the literature (see, e.g., Masae et al. 2019a).

Non-conventional warehouses differ from conventional warehouses with respect to the arrangement of picking aisles and cross aisles. They are often aimed to facilitate reaching certain areas of the warehouse and/or to improve space utilization (Masae et al. 2019a). Examples of non-conventional warehouse layouts include the fishbone (see, e.g., Gue and Meller 2009, Pohl et al. 2009), U-shaped (see, e.g., Glock and Grosse 2012, Glock et al. 2019), chevron (see, e.g., Öztürkoğlu et al. 2012, Masae et al. 2019b), and leaf and butterfly layout (see, e.g., Öztürkoğlu et al. 2012).

Warehouses can be further distinguished according to the number of depots, the number of storage locations assigned to an item type, and the width of picking aisles as follows:

- *Number of depots:* A warehouse either contains a single depot (centralized depositing) or multiple depots (decentralized depositing). In warehouses with decentralized depositing, the order pickers can drop off the retrieved items at multiple end depots, e.g., at the front and/or rear end of each picking aisle or at dedicated positions of a conveyor belt. Thus, unnecessary trips back to a central depot, which are the main disadvantage of traditional picker-to-parts setups, can be reduced.
- *Number of storage locations assigned to an item type:* An item type can be available from exactly one storage location (dedicated storage), or an item type can be assigned to more than one storage location (scattered storage). With scattered storage, the probability to have a requested item close-by the order picker handling the respective customer order can be increased, regardless of where the order picker is currently positioned in the picking area. In this way, unproductive walking of order pickers is reduced (Weidinger 2018). Therefore, it is not surprising that modern e-commerce warehouses of companies like Amazon or Zalando often apply scattered storage (Goeke and Schneider 2018) and that this setting is receiving increasing attention in the scientific literature (see, e.g., Goeke and Schneider 2018, Weidinger 2018, Boysen et al. 2019a). A disadvantage of scattered storage concerns the increase in time required to place incoming items into stock, which results from the fact that items of the same item type have to be transported to multiple storage locations (Weidinger and Boysen 2018).
- *Width of picking aisles:* Warehouses with standard, narrow, and wide picking aisles can be found in the literature and in practice. In standard picking aisles, order pickers can pass each other, and items can be retrieved from both sides of a picking aisles without performing additional movements (Scholz et al. 2017). If picking aisles are narrow, order pickers can retrieve items from both sides of a picking aisle without performing additional movements (Hong et al. 2012). Note that narrow picking aisles prohibit an order picker to pass another order picker in the same picking aisle, and congestion has to be considered in terms of waiting time (see, e.g., Parikh and Meller 2009, Hong et al. 2012). Contrary,

if picking aisles are wide, additional movements are required to retrieve items located on different sides of a picking aisle (Goetschalckx and Ratliff 1988).

2.3.2 Order picking process

Order picking results from incoming customer orders that require the retrieval of the requested items from their storage locations of a warehouse. In low-level picker-to-parts systems, order pickers use a picking device and a pick list (ten Hompel et al. 2011). A picking device (e.g., a picking cart or a roll cage) serves to transport the retrieved items through the warehouse. Pick lists are generated based on the incoming customer orders. Each pick list specifies a picking order and contains a non-empty set of order lines, where each order line indicates a particular item, the requested quantity of this item as well as its storage location in the picking area (ten Hompel et al. 2011, Henn and Wäscher 2012). While a customer order is associated with the requested items of a single customer, a picking order can contain (i) all requested items of one or more customer orders, (ii) some of the requested items of one or more customer orders, or (iii) a combination of (i) and (ii). The order lines of a pick list are sorted in the sequence in which the items are to be retrieved by an order picker. Pick lists can be provided in paper form or from the warehouse management system to electronic means of communication, such as handheld scanners, smartphones, or tablets (Koch 2014).

A typical order picking process in a picker-to-parts system is shown in Figure 2.3 and can be described as follows: An order picker starts at the depot, where she receives a pick list and a picking device, and walks or rides to the storage location of the first item specified by the pick list. When she arrives at this storage location, she retrieves the item in the requested quantity and places the item(s) onto the picking device. Subsequently, she either proceeds to the next storage location if the picking order is not completed, or she returns to the depot and hands over the picking device with the retrieved items if there are no further items to be picked. Consequently, each picking order is associated with an order picking tour that starts from the depot, proceeds along the storage locations of all items specified by the picking order, and ends at the depot (Koch 2014). Obviously, if there are further picking orders to be processed, she receives a new pick list and a picking device, and the described procedure is repeated.

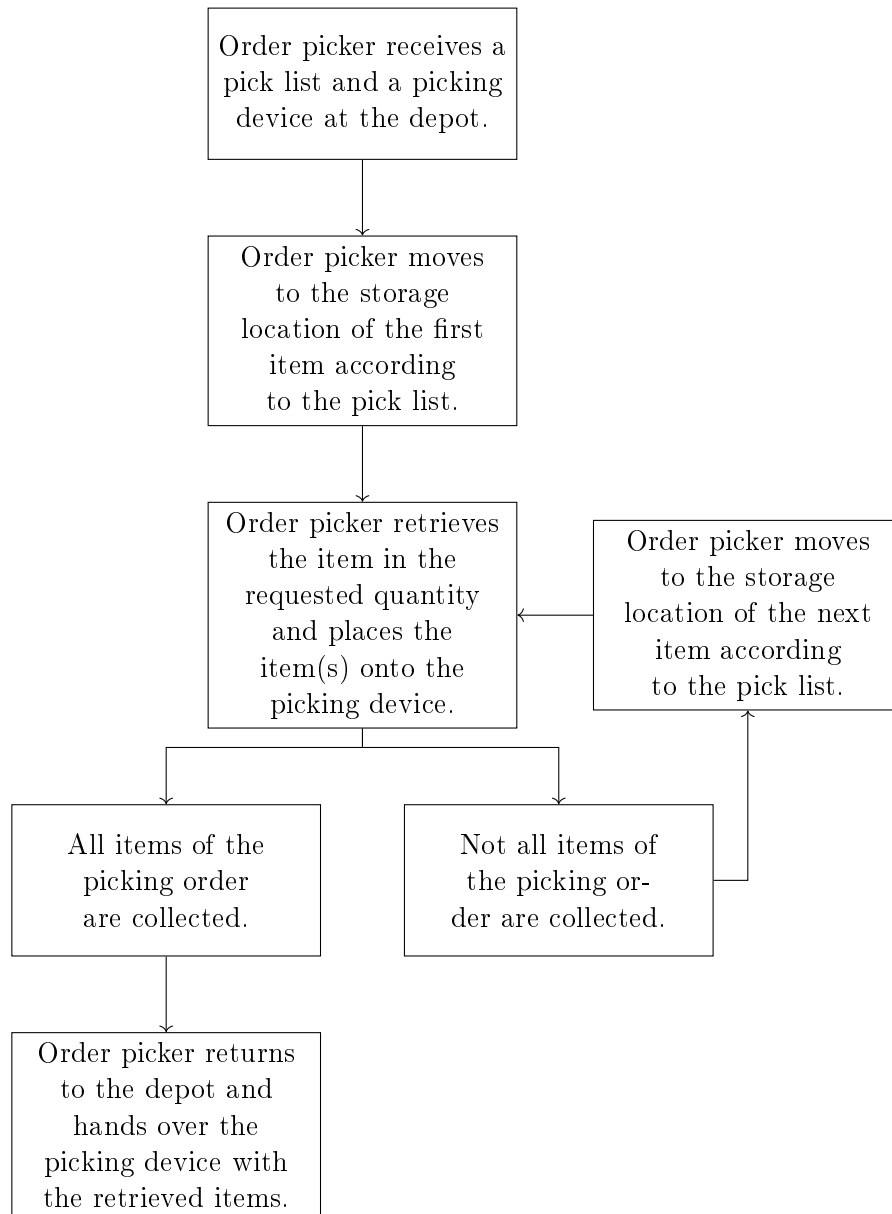


Figure 2.3: An example of an order picking process in a picker-to-parts system.

2.4 Common planning objectives in order picking

Common objectives in order picking include the maximization of the customer service level and the minimization of the (operational) cost (Goetschalckx and Ashayeri 1989, de Koster et al. 2007). The customer service level depends on delivery lead times and (contractually) agreed delivery dates. The delivery lead time describes the time that has elapsed between placing a customer order and the delivery to the customer. Note that other factors (e.g., completeness and correctness of the delivery) also affect the customer service level, but they are not relevant for the problems addressed in this dissertation.

A short delivery lead time can be achieved by a reduction of the order picking time, i.e., the time required for picking the requested items of a customer order, because it is an essential part of the delivery lead time of a customer order (Henn et al. 2010). It is therefore appropriate to analyze the integral components of order picking time, namely setup, travel, search, and pick time (Petersen 1999, Henn et al. 2012). Setup time is defined by the time for administrative and setup tasks (e.g., receipt of the pick list and picking device). Travel time is the time an order picker requires to travel from the depot to the first picking location, between the picking locations, and from the last visited picking location to the depot. The time an order picker spends on identifying the respective storage locations and the requested items is called search time. Pick time is defined by the time required to retrieve the requested items from their storage locations and to place them onto the picking device.

Figure 2.4 presents a typical distribution of order picking time among the described activities. As the figure shows, traveling is the most time-consuming activity with 50% of order picking time. According to Tompkins et al. (2010), the other components can either be considered to be constant (e.g., search and pick time) or negligible (e.g., setup time).

Consequently, minimizing total travel time, i.e., the time for picking a given set of customer orders, seems to be an appropriate lever for improving the customer service level. Moreover, a reduction of the total travel time affects the objective of minimizing the (operational) cost: In the short run, labor costs related to, e.g., the regular working time of the order pickers, overtime, or temporary workforce can be reduced (Wäscher 2004). In the long run, even the permanent workforce can decrease (Henn et al. 2010). Improving customer service on the one hand and reducing costs on the other hand, positively affect the competitiveness of the overall

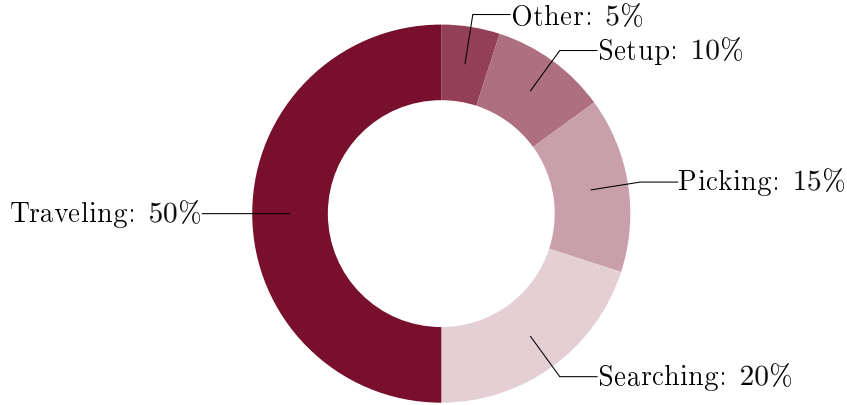


Figure 2.4: Typical distribution of order picking time (based on Tompkins et al. (2010)).

supply chain, and it is therefore not surprising that they often serve as primary objectives for warehouse managers.

Besides minimizing total travel time, the minimization of the total travel length is often used as planning objective in the order picking literature. Assuming that the order picker's travel velocity is constant and neglecting the order picker's acceleration and braking movements, the minimization of the total length of all order picking tours, which the order picker has to cover to collect the requested items of a given set of customer orders, is equivalent to the minimization of the total travel time (Jarvis and McDowell 1991, Henn et al. 2012). However, in order picking systems with high space utilization in which order picker blocking can occur, this assumption does not hold. In such systems, narrow picking aisles prohibit an order picker to pass another order picker in the same picking aisle, and congestion has to be considered in terms of waiting time (see, e.g., Parikh and Meller 2009, Hong et al. 2012).

As stated above, meeting contractually agreed or promised next or even same-day deliveries is another central component of the customer service level. Customer orders often have to be completed until given due dates (i) to avoid delays in the scheduled departure of trucks delivering the requested items to customers or (ii) to provide the input to a production system on time and thus to avoid production delays (Henn and Schmid 2013). In these contexts, tardiness-related objectives, e.g., the minimization of the tardiness of all customer orders, are often considered (see, e.g., Henn and Schmid 2013, Scholz et al. 2017). The tardiness of a customer order can be generally described as the extent to which the due date of a customer order is violated. Order picking time is an integral part of the tardiness, so minimizing

total travel time/length also has a positive effect on meeting due dates.

Because we consider different planning problems, different objectives (i.e., minimizing the total tour length and minimizing the tardiness of a given set of customer orders) are pursued in this dissertation.

2.5 Planning problems in order picking

Due to the large number of research papers dealing with planning problems in order picking, we cannot provide an exhaustive review of every research contribution. Instead, we review some outstanding papers related to the problems addressed in this dissertation.

2.5.1 Warehouse layout design

For the sake of completeness, we briefly describe the main problems of the warehouse layout design and mention relevant literature references although no decisions on the design of the warehouse layout are made in this dissertation.

In the context of order picking, the design of the warehouse layout concerns the facility layout of the order picking system and the layout within the order picking system (de Koster 2015). The facility layout deals with the question of where to locate the various warehouse areas, such as the receiving, storing, picking, sorting, packing, and shipping area. A common objective is to minimize handling cost, which is often represented by a linear function of the travel distance. Both Meller and Gau (1996) and Tompkins et al. (2010) give a review of the literature on the design of the facility layout.

The design of the layout within the order picking system, called internal layout design or aisle configuration problem, concerns the number of blocks and the number, length, and width of the picking and cross aisles. The minimization of the travel distance is again the most common objective. Works investigating the internal layout design are, for example, those of Caron et al. (1998, 2000), Petersen and Aase (2002), Roodbergen and Vis (2006), de Koster et al. (2007), and Roodbergen et al. (2008).

2.5.2 Item storage assignment

The item storage assignment problem belongs to the class of assignment problems which deal with the matching of two or more sets of elements (e.g., items, storage locations, machines, tasks) to each other (Pentico 2007). Generally speaking, the problem models the decision on how to assign items to storage locations of a warehouse such that a given performance measure (e.g., total tour length for collecting the items of a given set of customer orders) is optimal (Gu et al. 2007).

Besides a few exact algorithms (see, e.g., Hausman et al. 1976), the literature proposes various strategies to assign items to storage locations (see Figure 2.5). Frequently studied strategies of storage assignment are random (or chaotic), dedicated, and class-based storage (Gu et al. 2007, 2010, de Koster et al. 2007).

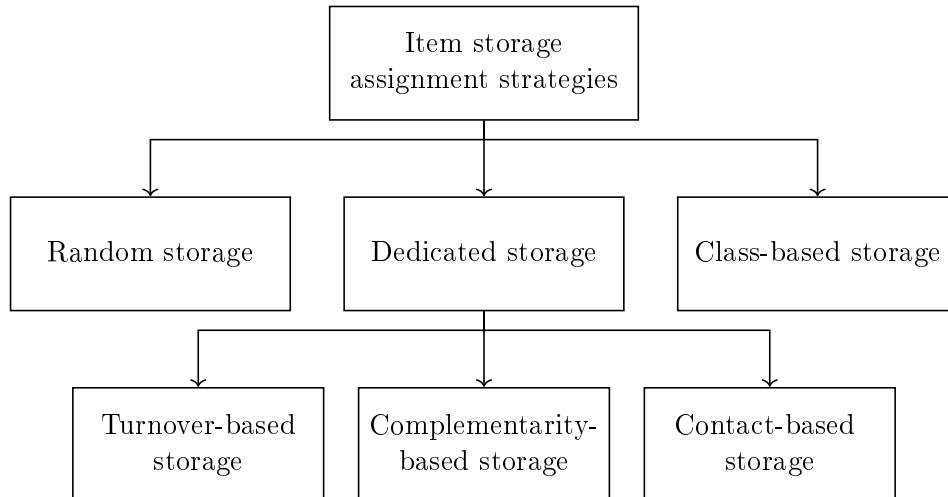


Figure 2.5: Overview of common item storage assignment strategies.

A random storage strategy arbitrarily assigns items to available, empty storage locations with equal probability (Petersen 1997). An advantage of this strategy is that it leads to a high storage space utilization (or a low space requirement) because storage locations are not reserved for items that are out of stock (see, e.g., de Koster et al. 2007). On the negative side, random storage results in long travel times if items jointly requested by customers are stored in storage locations that are far away from each other.

Contrary to random storage, dedicated storage assigns items to consistent storage locations for a relatively long period of time (Wäscher 2004). Dedicated storage has the advantage that order pickers become familiar with the storage locations

assigned to the different items over time. This speeds up order picking as compared to random storage. Dedicated storage is also advantageous if items differ according to their weight (de Koster 2015). For instance, to avoid work accidents caused by the lifting (removal) of heavy items to (from) relatively high rack positions, heavy items may be dedicated to lower storage locations in a rack. A drawback of dedicated storage is that it leads to a lower degree of storage space utilization because sufficient space is reserved for storing the maximum inventory level. Even if an item is out of stock, dedicated storage locations are not used for storing other items.

Approaches for assigning dedicated storage locations can be divided into three classes, depending on whether they are turnover-based, complementarity-based, or contact-based (see, e.g., Wäscher 2004, de Koster et al. 2007). The turnover-based approach considers the demand frequencies of items when allocating dedicated storage locations. The demand frequency indicates how often an item has been requested by customers within a certain period of time (Wäscher 2004). While items with high demand frequencies are located close to the depot, items with low demand frequencies are located farther away from the depot.

The cube-per-order index (COI), a modification of the turnover-based approach, does not only consider demand frequencies of items but also their space requirement (Heskett 1963). The COI is defined as the ratio of the item’s storage space requirement (cube) to its demand frequency per period. The idea of the COI is to store compact, frequently requested items (characterized by low COI values) close to the depot and to move bulky, rarely requested items (characterized by high COI values) to storage locations that are located far away from the depot (Wäscher 2004). A disadvantage of turnover-based storage assignment is that a change in demand frequencies implies a reassignment of items to storage locations. Moreover, the implementation of the concept of turnover-based storage requires considerably more information than random storage (de Koster 2015). Applications of the COI-based rule can be found in Kallina and Lynn (1976) and Malmborg and Bhaskaran (1987).

Contrary to turnover-based storage, complementarity-based and contact-based approaches do not only take into account the demand frequency of items when assigning storage locations but also consider whether different items show similarities in demand. In general, complementarity-based approaches distinguish two phases. In the first phase, items are clustered based on a complementarity measure that defines the strength of joint demand (de Koster 2015). A high complementarity value between two items indicates that these items are frequently requested together. The

clustering can be defined as a p -median problem (Rosenwein 1994). In the second phase, items are assigned cluster by cluster to storage locations as close to each other as possible (Wäscher 2004). To determine the storage locations to which the items of a cluster should be assigned, Liu (1999) suggests the following procedure: The item with the highest demand is assigned to the storage location closest to the depot. All other items of the same cluster are assigned to storage locations close to this item according to the turnover-based approach. A disadvantage of the complementarity-based approach is that an order picker does not necessarily travel directly between the storage locations of two items even if they are requested by the same customer.

Therefore, contact-based methods assign items to dedicated storage locations with respect to direct travels (called contacts) between items (Wäscher 2004). For contact-based approaches, a PRP has to be solved to determine the contacts for a given set of customer orders, which in turn means that the storage locations of the items must be known (see Section 2.5.3). Because a simultaneous solution of both problems is rather not realistic for instances of practically relevant size, van Oudheusden et al. (1988) propose a procedure which alternates between both problems.

The concept of class-based storage groups items into several classes, typically based on their demand frequency or the COI. These classes are then assigned to dedicated storage areas of the warehouse (see, e.g., Jarvis and McDowell 1991, Petersen and Schmenner 1999). Items with the highest demand frequency are stored close to the depot, and items with the lowest demand frequency are assigned to storage locations that are far away from the depot. Storage assignment within an area is random. Similar to the complementarity-based approach, it is necessary to solve a clustering problem.

A common variant of class-based storage is the ABC storage strategy, which is based on the Pareto principle according to which a few items are responsible for a large proportion of the total demand (de Koster et al. 2007). ABC storage groups items into three classes (A, B, and C), with class A representing fast moving items, class B including items with medium demand, and class C covering the least requested items. Other groupings of items (e.g., into more than three classes) are possible and may give additional gains with respect to the objective of minimizing the total tour length for collecting the items of a given set of customer orders (Roodbergen 2001, de Koster 2015). An advantage of class-based storage is that it leads to short travel times because fast moving items are stored close to the depot. Addi-

tionally, high storage space utilization can be achieved because items are randomly assigned to storage locations within an area.

Note that random and dedicated storage represent extreme cases of the class-based storage strategy. In the case that each item belongs to a separate class, the class-based strategy corresponds to the dedicated storage strategy. In the other extreme case of a single class, the class-based strategy corresponds to the random strategy.

With respect to item storage assignment strategies, studies mainly focus on random storage. Analytical models which can be used for dedicated or class-based storage are missing (see, e.g., de Koster 2015). Also, item-specific characteristics (e.g., weight, hazardousness, and temperature requirements) are often neglected in the literature when assigning items to storage locations (Dekker et al. 2004). Moreover, the impact of storage assignment on the performance of a picker routing strategy is hardly studied. Instead, when discussing the performance of a picker routing strategy, random storage is usually assumed (de Koster 2015).

2.5.3 Picker routing

In this section, we first describe the standard single picker routing problem (standard SPRP). We then give an overview of the solution methods for the standard SPRP and its variants. For an extensive literature review on picker routing, we refer the reader to Masae et al. (2019a).

2.5.3.1 The standard picker routing problem

In general, PRPs aim at determining a cost-minimal order picking tour along the storage locations defined by a picking order (see, e.g., Ratliff and Rosenthal 1983). The standard SPRP is the most well-studied PRP in the literature. It can be defined as follows: Given a single-block parallel-aisle warehouse with a central depot and dedicated storage, the standard SPRP models the decision on how to route a single order picker through the warehouse (starting from and ending at the depot) such that a given performance measure (e.g., travel time or travel distance) for collecting the items defined by a picking order from their storage locations is optimal. The standard SPRP assumes that an order picker can retrieve items from both sides of a picking aisle without performing additional movements. Furthermore, the capacity of the picking device used to transport the items through the warehouse is assumed to be sufficient for carrying all items contained in the picking order.

PRPs classify as a special case of the traveling salesman problem (TSP), and thus, TSP approaches can be used to solve a PRP. However, order picking tours that are performed by order pickers in a single- or multi-block parallel-aisle warehouse exhibit a specific structure, which is not considered by general TSP formulations. For example, to cross over from one picking aisle to another picking aisle, an order picker uses one of the cross aisles. Problem-specific solution approaches taking specific characteristics into account may therefore outperform TSP approaches with respect to the size of the instances that can be solved and the runtimes that are required to solve these instances (see, e.g., Scholz et al. 2016, Goeke and Schneider 2018).

Problem-specific solution approaches for the standard SPRP can be distinguished into exact algorithms, construction heuristics, and metaheuristics.

Exact solution approaches In a seminal work, Ratliff and Rosenthal (1983) present a graph-based dynamic programming algorithm to solve the standard SPRP to optimality. The time complexity of their algorithm is linear in the number of picking aisles. Scholz et al. (2016) propose a graph-based mathematical model formulation that considers specific properties of optimal order picking tours of the standard SPRP. For example, they show that it is sufficient that each picking aisle is only represented by six picking locations instead of considering all picking locations. Their approach is compared to three TSP formulations and one Steiner TSP formulation. The authors demonstrate that their formulation outperforms these general formulations with respect to the size of instances that can be solved and the corresponding runtimes. Pansart et al. (2018) introduce an exact dynamic programming approach and a mixed integer linear programming formulation, which is based on a single-commodity flow formulation of the Steiner TSP. Goeke and Schneider (2018) propose a compact formulation that directly exploits the property of an optimal order picking tour in which two consecutive picking aisles can only be connected using four possible configurations as presented in Ratliff and Rosenthal (1983). Additionally, their formulation is not based on classical subtour elimination constraints. In numerical studies, Goeke and Schneider (2018) show that using their formulation, large instances can be solved within short runtimes. On a set of benchmark instances with up to 30 picking aisles and 45 required picking locations, their formulation clearly outperforms that of Scholz et al. (2016) and is about six times faster than the one of Pansart et al. (2018). Although using the formulation

of Goeke and Schneider (2018) to solve the standard SPRP cannot compete with the performance of the algorithm proposed by Ratliff and Rosenthal (1983), it is advantageous because it can be solved with a mathematical programming solver, so neither knowledge of a higher programming language nor experience in algorithmic programming are required (Goeke and Schneider 2018).

Construction heuristics In practice, the standard SPRP is often solved by construction heuristics because the resulting order picking tours follow straightforward and non-confusing patterns compared to an optimal order picking tour (de Koster et al. 1999). Hence, the risk of not collecting a requested item during order picking can be reduced (Petersen and Schmenner 1999). The most common heuristic picker routing strategies include the S-shape (or traversal) strategy by Goetschalckx and Ratliff (1988), the return, midpoint, and largest gap strategy by Hall (1993), and the composite strategy by Petersen (1995). In Figure 2.6, we give an example of the resulting order picking tours (starting from and ending at a central depot) through a single-block parallel-aisle warehouse of an order picker applying different picker routing strategies to collect a given picking order. The black rectangles represent picking locations of the customer order.

The applied heuristic picker routing strategies can be described as follows:

- *S-shape*: According to the S-shape strategy, the order picker starts at the depot, proceeds to the leftmost picking aisle that contains at least one picking location and traverses it completely. Then, the order picker enters all other picking aisles alternately from the rear cross aisle and the front cross aisle (if they contain at least one picking location) and traverses them completely. An exception may occur in the last picking aisle to be visited: if the order picker enters this picking aisle from the front cross aisle, she travels to the last item to be picked in the picking aisle, returns to the front cross aisle and from there to the depot (Hall 1993).
- *Return*: The return strategy proposes that each picking aisle containing an item to be picked is entered and left from the same cross aisle. This strategy is often used in warehouses in which closed-end picking aisles occur, i.e., the order picker can only use the same cross aisle to travel from one picking aisle to another picking aisle (Roodbergen and de Koster 2001a).
- *Midpoint*: Following the midpoint strategy, picking aisles are entered as far as

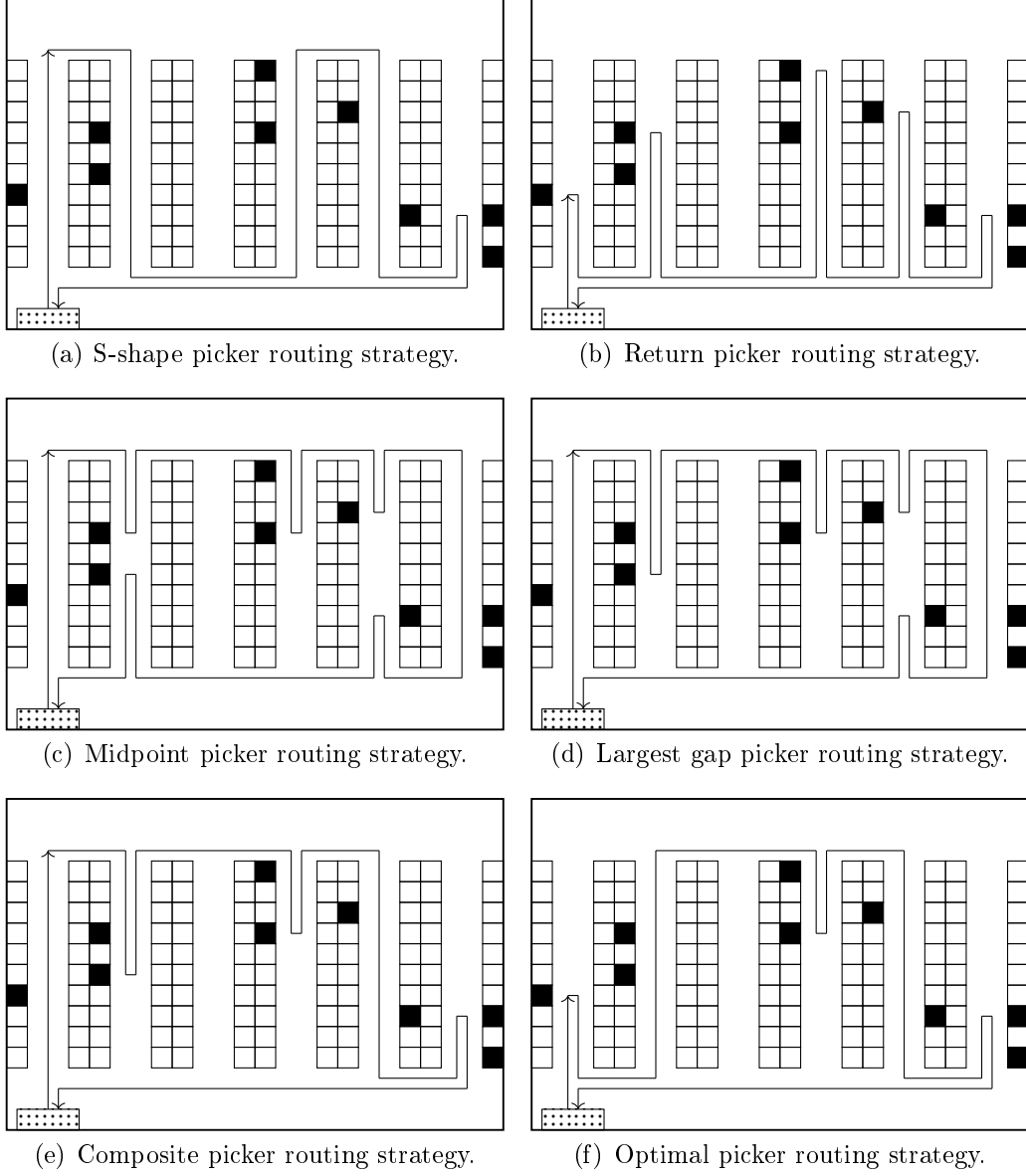


Figure 2.6: Example of the resulting order picking tours for different picker routing strategies. The figure illustrates the resulting picking tours (starting from and ending at the depot) through a single-block parallel-aisle warehouse of an order picker applying different picker routing strategies to collect a given picking order. The black rectangles represent picking locations of the customer order.

the midpoint of a picking aisle from the front and rear of a cross aisle (Hall 1993). Thus, the order picker performs either a return route from the front cross aisle, a return route from the rear cross aisle if the items are above the midpoint, or return routes from both the front and the rear cross aisle (Petersen and Schmenner 1999). The leftmost picking aisle (which contains an item to be picked) is traversed to enter the rear cross aisle and the rightmost picking aisle (which contains an item to be picked) to enter the front cross aisle.

- *Largest gap*: In the case of the largest gap strategy, the order picker traverses the leftmost and the rightmost picking aisle completely if they contain at least one picking location. Other picking aisles that contain an item to be picked are entered in such way that the largest gap of a picking aisle is not traversed. A gap defines the distance in a picking aisle between (i) any two adjacent picking locations, (ii) the closest picking location (to the front cross aisle) and the front cross aisle, or (iii) the farthest picking location (to the front cross aisle) and the rear cross aisle. The largest gap is the part of a picking aisle that the order picker does not traverse. If the largest gap corresponds to (i), the order picker enters and returns via the same front (rear) cross aisle. If the largest gap corresponds to (ii) or (iii), the order picker enters and returns either from the rear (ii) or front cross aisle (iii). With respect to the objective of minimizing the total tour length for collecting the items given by a picking order, the largest gap strategy always performs better than or at least equal to that of the midpoint strategy (Hall 1993).
- *Composite*: The composite strategy includes a dynamic programming component which determines for each picking aisle whether to use an S-shape or a return strategy, depending on the picking locations in the next picking aisle. For example, consider a picking aisle for which the shortest travel time could be achieved if the picking locations are visited using the return strategy. However, the total travel time could be reduced if the picking aisle is completely traversed (S-shape strategy) because this could allow a better starting point for the next picking aisle (Petersen 1995, Roodbergen and de Koster 2001a).

Metaheuristic solution approaches To the best of our knowledge, de Santis et al. (2018) are the only ones who propose a metaheuristic solution approach for the (standard) SPRP. The authors develop an adapted ant colony optimization algorithm for a parallel-aisle warehouse layout with an arbitrary number of blocks.

In the context of order picking, metaheuristic solution approaches have been used to solve the standard SPRP or variants of the standard SPRP integrating other planning problems such as item storage assignment or order batching (see, e.g., Tsai et al. 2008, Chen et al. 2015). For a review on metaheuristic solution approaches combining PRPs with other planning problems, the reader is referred to van Gils et al. (2018).

2.5.3.2 Variants of picker routing problems

Variants of PRPs include different warehouse layouts, decentralized depositing, arbitrary start and end locations of an order picking tour, scattered storage, decoupling of order picker and picking cart, and precedence constraints. Such variants are briefly outlined in the following.

Different warehouse layouts Roodbergen and de Koster (2001a) and Masae et al. (2020) present extensions of the exact algorithm proposed by Ratliff and Rosenthal (1983) to address a two-block parallel-aisle warehouse layout. The exact approaches proposed by Scholz et al. (2016) and Pansart et al. (2018) for solving the standard SPRP can be also applied to deal with multi-block parallel-aisle warehouse layouts. Çelik and Süral (2014) provide a polynomial-time algorithm for a fishbone warehouse layout. Masae et al. (2019b) introduce an exact algorithm based on dynamic programming for the chevron warehouse layout and modify the midpoint and largest gap picker routing strategies to address their scenario.

Heuristic approaches for multi-block parallel-aisle warehouse layouts can be found in Vaughan (1999), Roodbergen and de Koster (2001b), Theys et al. (2010), and Çelik and Süral (2019). Vaughan (1999) propose a so-called aisle-by-aisle heuristic, which is based on dynamic programming. Roodbergen and de Koster (2001b) adapt existing picker routing strategies (S-shape, largest gap) and introduce a new picker routing strategy, called combined heuristic. Theys et al. (2010) use the Lin-Kernighan-Helsgaun heuristic for routing order pickers. A graph theory-based heuristic is proposed in Çelik and Süral (2019).

Decentralized depositing The algorithm of Ratliff and Rosenthal (1983) is extended in de Koster and van der Poort (1998) to the case of multiple end depots assuming a single-block parallel-aisle warehouse. Items can be dropped off anywhere along the front cross aisle. Consequently, the start and end locations of an order

picking tour can be anywhere along the front cross aisle. Scholz et al. (2016) outline how their model formulation for the standard SPRP can be extended to cope with decentralized depositing of items at the front or the rear end of each picking aisle. In the mathematical model formulation of Goeke and Schneider (2018), it is assumed that an order picker can select an arbitrary end depot from a set of possible candidate depot locations to drop the items.

Arbitrary start and end locations of an order picking tour Löffler et al. (2020) extend the algorithm of Ratliff and Rosenthal (1983) by allowing arbitrary start and end locations of an order picking tour for a single-block parallel-aisle warehouse layout and Masae et al. (2020) for a two-block parallel-aisle warehouse layout. Masae et al. (2020) also propose a heuristic picker routing strategy, denoted as S*-shape, and compare the performance of S*-shape to those of the exact picker routing strategy.

Scattered storage Daniels et al. (1998) propose a TSP formulation for a PRP in which any item can be available from multiple storage locations for an arbitrary warehouse layout and compare several heuristic solution approaches. For a single-block parallel-aisle warehouse, scattered storage is also investigated in Weidinger (2018) and Goeke and Schneider (2018), who extend their mathematical model formulation for the SPRP described above. Weidinger (2018) proposes three different routing heuristics based on the algorithm of Ratliff and Rosenthal (1983) to solve the SPRP with scattered storage. Additionally, he provides a proof of NP-hardness in the strong sense. Goeke and Schneider (2018) show that their formulation outperforms those of Weidinger (2018) with respect to solution quality and runtimes.

Decoupling of order picker and picking cart Goeke and Schneider (2018) are the first to propose a mathematical model formulation to cope with the decoupling of order picker and picking cart. They assume that order pickers are allowed to park the picking cart during an order picking tour, retrieve a few items walking on their own, return to the picking cart and continue the order picking tour with the picking cart.

Precedence constraints In practice, the routing of order pickers is often subject to precedence constraints (Chabot et al. 2017). These constraints define that certain items need to be collected before other items due to fragility, stackability,

shape and size, and preferred unloading sequence. Variants of PRPs considering precedence constraints can be found in the following papers. Dekker et al. (2004) examine combinations of item storage assignment strategies and heuristic picker routing strategies for a real-world application arising in a warehouse of a wholesaler of tools and garden equipment. The picking area is characterized by three blocks, closed-end picking aisles, and two floors. Arbitrary start and end locations of an order picking tour are allowed. Furthermore, a guideline requiring that breakable items have to be picked after unbreakable items to prevent damaging to the breakable items has to be considered. To address this requirement, the authors propose to assign breakable items to storage locations in such a way that the requirement is automatically met. For example, breakable items are stored in the rightmost picking aisle, and with the start location being at the leftmost picking aisle, the requirement can be automatically met.

Chabot et al. (2017) introduce the so-called order picking problem under weight, fragility and category constraints for a single-block parallel-aisle warehouse. These precedence constraints can be defined as follows: As soon as the total weight of the already collected items exceeds a threshold weight value, heavy items can no longer be picked, and the order picker may only collect light items (weight constraints). Thus, another order picking tour is necessary if not all heavy items have been collected. To avoid damage to fragile items, heavy items must not be placed on top of fragile items. The authors refer to this restriction as fragility constraint. The category constraints define that non-food items have to be placed underneath food items on the pallet to avoid contamination. The authors propose a capacity-indexed mathematical model formulation and a two-index vehicle-flow formulation. To solve the problem, four heuristics (S-shape, largest gap, mid-point, and ALNS) are presented. Furthermore, a branch-and-cut algorithm and cutting planes are applied to solve the two formulations of the problem considering the precedence constraints.

In Table 2.1, we give an overview of the literature described above. The references are sorted in ascending order of the year of publication and alphabetically by the surname of the first author. The first column includes the respective reference. The next seven columns specify characteristics of the picking area, i.e., the arrangement of picking and cross aisles (single-block, two-block, three-block, multi-block, or other warehouse layouts), decentralized depositing (otherwise the reference assumes a central depot), and arbitrary start and end points of an order picking tour. Columns nine and ten indicate whether scattered storage assignment (otherwise the reference

assumes a dedicated storage assignment) and decoupling of order picker and picking cart are considered. Column eleven specifies whether precedence constraints are taken into account. Finally, the last three columns classify the references according to the solution approach (exact, construction heuristic, and metaheuristic). Whenever a reference fulfills one of the described characteristics, this is indicated by a bullet point in the corresponding cell of the table.

Reference	Single-block warehouse layout	Two-block warehouse layout	Three-block warehouse layout	Multi-block warehouse layout	Other warehouse layout	Decentralized depots	Arbitrary start and end points	Scattered storage	Decoupling of picker and cart	Precedence constraints	Exact solution method	Construction heuristic	Metaheuristic solution method
Ratliff and Rosenthal (1983)	•										•		
Goetschalckx and Ratliff (1988)	•											•	
Hall (1993)	•											•	
Petersen (1995)	•											•	
Daniels et al. (1998)					•			•					•
De Koster and van der Poort (1998)	•					•					•		
Vaughan (1999)				•								•	
Roodbergen and de Koster (2001a)		•									•		
Roodbergen and de Koster (2001b)				•								•	
Dekker et al. (2004)			•							•		•	
Theys et al. (2010)				•								•	
Çelik and Süral (2014)					•						•		
Scholz et al. (2016)				•		•					•		
Chabot et al. (2017)	•									•	•	•	•
De Santis et al. (2018)				•									•
Goeke and Schneider (2018)	•					•		•	•		•		
Löffler et al. (2020)	•						•				•		
Pansart et al. (2018)				•							•		
Weidinger (2018)	•							•				•	
Çelik and Süral (2019)				•								•	
Masae et al. (2019b)					•						•		
Masae et al. (2020)		•					•				•		

Table 2.1: Overview of single picker routing problems.

To summarize, our survey of the literature shows that the standard SPRP is the most well-studied PRP. To solve the standard SPRP, mainly exact solution methods have been proposed. This can be explained by the fact that the specific characteristics of the problem (e.g., rectangular warehouse layout, parallel picking aisles) make it possible to develop algorithms that obtain optimal solutions for practically relevant instance sizes within shortest runtimes. Variants of the standard SPRP mainly deal with different warehouse layouts, multiple end depots, arbitrary start and end locations of an order picking tour, and/or scattered storage. Although precedence constraints in order picking arise in many real-world applications, works considering precedence constraints are rather rare.

2.5.4 Order batching

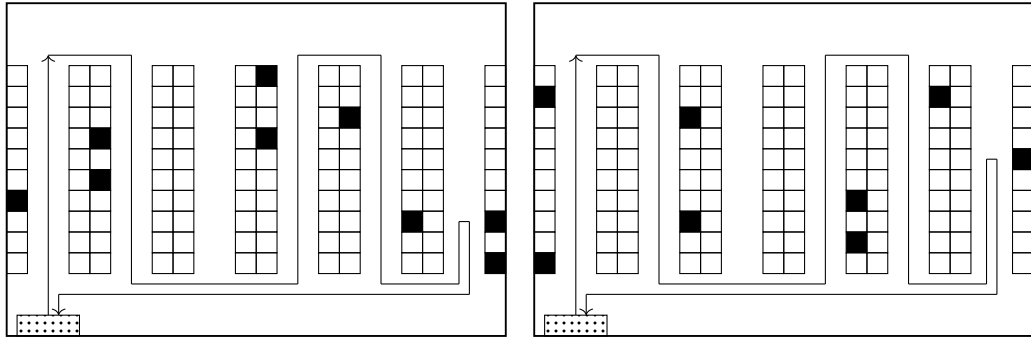
In this section, we first outline fundamentals of order batching. Then, we define the standard OBP and give a comprehensive review of the state-of-the-art solution methods to address the standard OBP. Furthermore, we detail the literature on OBP variants.

2.5.4.1 Introduction

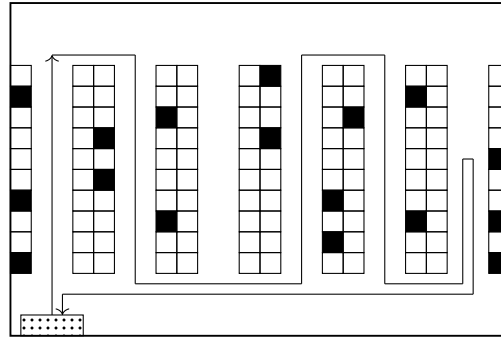
If the number of items per customer order is large in relation to the capacity of the picking device, customer orders are usually picked individually on a single order picking tour. This way of picking is called single order picking, pick-by-order, or discrete order picking. However, if the number of items per customer order is small, the efficiency of the order picking process can be increased by consolidating a set of customer orders into a single order picking tour. Obviously, picking multiple customer orders on a single order picking tour increases the pick density per tour. This order picking method is referred to as pick-by-batch or order batching (de Koster et al. 1999).

In Figure 2.7, we illustrate an example for reducing the total tour length required to collect the items of two customer orders by order batching in a single-block parallel-aisle warehouse. The order picker is sequenced by the S-shape routing strategy. The black rectangles represent the picking locations of the requested items, and the solid line indicates the order picking tour through the warehouse, which starts from and ends at the central depot. Figures 2.7(a) and 2.7(b) depict the resulting order picking tours for collecting the items of two customer orders on two separate

order picking tours. Figure 2.7(c) demonstrates the resulting single order picking tour if the customer orders are grouped into a batch. The figures show that the total tour length can be reduced by simultaneously picking both customer orders on a single order picking tour because the tour length in Figure 2.7(c) is shorter than the sum of the tour lengths resulting from the order picking tours in Figure 2.7(a) and Figure 2.7(b).



(a) Resulting order picking tour if customer order A is picked individually. (b) Resulting order picking tour if customer order B is picked individually.



(c) Resulting order picking tour if customer orders A and B are picked simultaneously on a single order picking tour.

Figure 2.7: An example for reducing the total tour length by order batching assuming the S-shape picker routing strategy in a single-block parallel-aisle warehouse. The black rectangles represent the picking locations of the requested items, the solid line indicates the order picking tour through the warehouse, starting from and ending at the central depot.

A batch can contain (i) all requested items of one or more customer orders, (ii) some of the requested items of one or more customer orders, or (iii) a combination of (i) and (ii). The size of a batch is restricted by the capacity of the picking device and is often defined by the number of customer orders (see, e.g., Le-Duc and de Koster 2007) or the number of items (see, e.g., Bozer and Kile 2008).

There are basically two principles for batching customer orders: proximity batching and time window batching (Choe and Sharp 1991). Proximity batching groups customer orders based on the proximity of their storage locations to those of other customer orders, i.e., customer orders whose item storage locations are close together in the warehouse are combined. Time window batching can be carried out as fixed or variable time window batching (van Nieuwenhuyse and de Koster 2009). In fixed time window batching, customer orders arriving during a predetermined (fixed) time interval are grouped into a batch. When batching with a variable time window, a specified number of customer orders is assigned to a batch.

With respect to the availability of information about the customer orders, two situations can be distinguished: static and dynamic order batching (Yu and de Koster 2009). In static order batching, all customer orders and their composition, i.e., the requested items and corresponding demand quantities, are known at the beginning of the planning period. In dynamic order batching, information about the customer orders becomes available over time (see, e.g., Henn 2012).

A disadvantage of order batching is that the batched customer orders have to be separated before shipping to the respective customers. Two strategies exist that define when to separate the customer orders of a batch, namely the pick-and-sort and the sort-while-pick strategy. In the case of the pick-and-sort strategy, customer orders are separated at the depot after the order picking process. According to the sort-while-pick strategy, customer orders are already separated during the order picking process. Consequently, no or only little sorting effort is necessary (see, e.g., van Nieuwenhuyse and de Koster 2009). To apply the sort-while-pick strategy, the picking device has to be equipped with separate bins for the individual customer orders (Gademann and van de Velde 2005). Note that if the items of a customer order are assigned to different batches, i.e., the items are collected on different order picking tours, additional effort is required to consolidate the items by customer orders at the end of the order picking process.

2.5.4.2 The standard order batching problem

Among various OBPs discussed in the literature, the standard OBP (as defined in Chapter 1) is the most well-studied OBP. Solution approaches for the standard OBP can be distinguished into exact algorithms, construction heuristics, and metaheuristics.

Exact solution approaches Gademann and van de Velde (2005) define the standard OBP as a generalized set partitioning problem and propose a branch-and-price algorithm with column generation for solving the linear programming relaxation of the problem. They compute the travel time required for collecting the items of a picking order by applying the exact algorithm of Ratliff and Rosenthal (1983). Problem instances with up to 32 customer orders, ten items per customer order, and a picking device capacity of four customer orders are solved to optimality in approximately six minutes on average. Bozer and Kile (2008) propose a mixed integer programming approach based on the S-shape picker routing strategy and solve instances with up to 25 customer orders, ten items per customer order, and a picking device capacity of 25 items to optimality. Their S-shape strategy is different from the originally proposed strategy because it does not consider the case in which the number of traversals is odd. In this case, the order picker enters the rightmost picking aisle from the front cross aisle, travels the picking aisle to the last item to be picked, returns to the front cross aisle and from there to the depot. Muter and Öncan (2015) develop a column generation approach using the S-shape, return, and midpoint picker routing strategy. They employ upper and lower bounding procedures that are strengthened by adding subset-row inequalities. With their set partitioning formulation, a small percentage of problem instances with up to 100 customer orders and an average number of six items per customer order for a picking device capacity of 24 items can be solved. Öncan (2015) introduce three mixed integer programming formulations considering the S-shape, return, and midpoint picker routing strategy. Optimal solutions are obtained for small-sized instances with 20 customer orders within a given runtime limit of three hours.

Construction heuristics Simple construction heuristics suggested for the standard OBP include priority rule-based, seed, and savings algorithms.

- *Priority rule-based algorithms*: Algorithms based on priority rules assign customer orders to batches in the sequence of non-ascending priority values ensuring that the picking device capacity is not violated. Examples of priority rules are the first-come first-serve (FCFS) rule and space-filling curves (Gibson and Sharp 1992, Pan and Liu 1995). While respecting the priority values, a customer order can be assigned to a batch either by the well-known next-fit, first-fit, or best-fit rule (see, e.g., Wäscher 2004).

- *Seed algorithms*: Elsayed (1981) and Elsayed and Stern (1983) introduced seed algorithms, which group customer orders into batches in a two-phase procedure, namely the seed selection and the order congruency phase. In the seed selection phase, a customer order (seed) is selected from a set of not yet assigned customer orders to form a batch. For example, the seed can be (i) a random customer order, (ii) the customer order with the smallest (largest) number of items, or (iii) the customer order with the largest number of picking aisles to be visited. Moreover, the seed can be defined in a single or in a cumulative mode. In the single mode, the first customer order assigned to a batch defines the seed. Contrary to this, in the cumulative mode, all customer orders included in a batch serve as the seed (Henn et al. 2012).

Subsequently, in the order congruency phase, unassigned customer orders are sequentially added to the batch as long as the picking device capacity is not exceeded. If the remaining capacity of the batch is not sufficient to add another customer order, a new batch is created, and the described procedure is repeated. With respect to the assignment of customer orders to a batch, a measure of proximity to the seed customer order of the batch is used. For instance, the customer order which has the largest number of identical picking locations with the seed is added to the batch. An overview of seed selection and order congruency rules is given in Ho et al. (2008).

- *Savings algorithms*: The so-called savings algorithms are based on the algorithm of Elsayed and Unal (1989) proposed for the vehicle routing problem. The classic savings algorithm, referred to as C&W(i) algorithm, computes the tour length saving that may result by picking two customer orders simultaneously on a single order picking tour instead of two separate tours for each feasible (with respect to the picking device capacity) pair of customer orders. Then, the pairs of customer orders are sorted in non-ascending order of the tour length saving. The algorithm starts with the pair of customer orders with the largest tour length saving. Three different cases may occur while assigning customer orders to a batch: (i) a new batch is created if neither of the two customer orders is yet assigned, (ii) if one of the customer orders is already assigned, the other customer order is assigned to the same batch if the picking device capacity is sufficient, (iii) the next pair of customer orders is considered if both of the customer orders are already assigned to batches or if the capacity

is not sufficient (Henn and Wäscher 2012).

Compared to C&W(i), tour length savings are recalculated after each assignment of customer orders to a batch in an extended variant of C&W(i), called C&W(ii) algorithm (Elsayed and Unal 1989).

Other construction heuristics are based on cluster analysis (see, e.g., Hwang and Kim 2005) and data mining approaches (see, e.g., Chen and Wu 2005).

Metaheuristic solution approaches Gademann and van de Velde (2005) present an iterated descent algorithm, in which the initial solution is generated by the FCFS rule. Neighbor solutions are obtained by interchanging two customer orders from two batches. The authors follow a first improvement strategy according to which the first improving neighboring solution is accepted as the current solution. In numerical studies, iterated descent is compared to their branch-and-price algorithm described above. For small-sized instances, iterated descent is able to provide optimal solutions. For larger instances with up to 32 customer orders, near-optimal solutions are found.

Albareda-Sambola et al. (2009) batch customer orders applying a variable neighborhood search algorithm with six local exchange schemes and three kinds of neighborhoods with different size. They compare their algorithm to FCFS, C&W(i), C&W(ii), and several seed algorithms using the S-shape, largest gap, and composite picker routing strategy. Problem instances with up to 250 customer orders and 36 items per customer order are considered. Contrary to many other publications in which the capacity of the picking device is limited by the number of items or the number of customer orders, Albareda-Sambola et al. (2009) define a total weight value that must not be exceeded when batching customer orders. They assume that an item has a weight of one, two, or three weight units. The authors show that their algorithm consistently finds better solutions compared to the construction heuristics.

Henn et al. (2010) propose an iterated local search and an ant colony optimization algorithm to solve the standard OBP. The S-shape picker routing strategy and the largest gap picker routing strategy are used for routing the order picker. Problem instances with up to 60 customer orders, 25 items per customer order, and a picking device capacity of 75 items are considered. With respect to solution quality, the authors demonstrate that both approaches provide improved solutions compared to several construction heuristics and to the iterated descent of Gademann and van de Velde (2005).

Henn and Wäscher (2012) design an attribute-based hill climber (ABHC) and a TS algorithm. Initial solutions are generated either with FCFS or C&W(ii). Their ABHC applies a set of attributes to overcome local minima and a set of moves to guide the search towards new solutions, where an attribute characterizes a solution feature. The first attribute set describes each solution of the problem by pairs of customer orders that are assigned to the same batch. The second attribute set refers to the assignment of customer orders to batches. Consider an example where customer orders o_1 and o_2 are assigned to batch b_1 , and customer orders o_3 and o_4 are assigned to batch b_2 . According to the second attribute set, this solution can be described by the attributes (b_1, o_1) , (b_1, o_2) , (b_2, o_3) , and (b_2, o_4) . For both algorithms, the neighborhood of a current solution is examined by applying a shift operator, a swap operator, and a combination of both. The shift operator generates a neighborhood by shifting a customer order from one batch to another batch and the swap operator by swapping two customer orders between two batches. The TS uses two variants for the exploration of the neighborhood. First, a best improvement (BI) strategy which explores the entire neighborhood and selects the best non-tabu solution with shortest tour length as next solution. Second, to reduce computation time, they implement an aspiration plus (AP) criterion that explores only a limited subset of the neighborhood of the current solution. The algorithm generates neighboring solutions until a solution is found that has a total tour length that is at most 5% longer than that of the current solution. Then, between $3 \cdot n$ and $5 \cdot n$ additional non-tabu solutions are generated, where n is the number of customer orders. Finally, the algorithm selects the solution with shortest tour length as next solution. The computation of the total tour length is based on the S-shape and largest gap picker routing strategy. Performance is evaluated in several computational studies benchmarking the ABHC and TS to C&W(ii) and to three local search algorithms proposed by Henn et al. (2010). Problem instances with up to 100 customer orders, 25 items per customer order, and a picking device capacity of 75 items are considered.

Chirici and Wang (2014) present two population-based methods, namely an item-oriented genetic algorithm (IGA) and a group-oriented genetic algorithm (GGA). Order pickers are routed according to the S-shape strategy. Their algorithms are tested on problem instances with up to 60 customer orders, 25 items per customer order, and a picking device capacity of 75 items. They demonstrate that both algorithms improve the objective function value compared to C&W(ii). In terms of

solution quality, GGA performs better than IGA at a similar runtime.

Öncan (2015) introduce an iterated local search with tabu thresholding (ILST) as intensification procedure. They use the same neighborhood structures as defined in Henn et al. (2010) and Henn and Wäscher (2012). The authors consider test instances with up to 100 customer orders, 25 items per customer order, and a picking device capacity of 75 items.

Koch (2014) and Koch and Wäscher (2016) suggest different genetic algorithms and assess their performance on problem instances with up to 80 customer orders, 25 items per customer order, and a picking device capacity of 75 items.

Hong and Kim (2017) propose a mixed integer programming approach based on the S-shape picker routing strategy and solve randomly generated instances with up to 500 customer orders. The authors compare their method to several lower bounds and construction heuristics. Problem instances with up to 500 customer orders, 2.02 items per customer order (defined by a density function), and a picking device capacity of 20 items are considered. For the largest-sized instances, their solutions are approximately equal to those obtained by C&W(ii).

Detailed results on the performance of the algorithms of Henn and Wäscher (2012), Öncan (2015), Koch (2014), and Koch and Wäscher (2016) are given in Chapter 3 because we compare our ALNS×TS to these algorithms. To the best of our knowledge, Henn and Wäscher (2012) and Öncan (2015) present the best performing metaheuristics for the standard OBP.

In Table 2.2, we summarize the exact and metaheuristic solution approaches to the standard OBP described above. The references are sorted in ascending order of the year of publication and alphabetically by the surname of the first author. The first column includes the respective reference. The second column specifies the solution approach(es), and the third column details the size of the problem instances used in the respective reference. The problem instance size is given by three values, where the first value describes the maximum number of customer orders, the second value denotes the maximum number of items per customer order, and the third value represents the maximum picking device capacity indicated by the number of customer orders, item units, or weight units.

In the following, we briefly summarize the main findings of our literature review on the standard OBP:

- Only a few exact solution approaches have been proposed in the literature to

Reference	Solution approach(es)	Problem instance size
Gademann and van de Velde (2005)	iterated descent and branch-and-price algorithm with column generation	32/10/4 customer orders
Bozer and Kile (2008)	mathematical model based on the S-shape strategy	25/10/25 item units
Albareda-Sambola et al. (2009)	variable neighborhood search	250/36/150 weight units
Henn et al. (2010)	iterated local search and ant colony optimization	60/25/75 item units
Henn and Wäscher (2012)	attribute-based hill climber and tabu search	100/25/75 item units
Chirici and Wang (2014)	item-oriented and group-oriented genetic algorithm	60/25/75 item units
Koch (2014)	genetic algorithms	80/25/75 item units
Muter and Öncan (2015)	column generation	100/10/48 item units
Öncan (2015)	mathematical models based on the S-shape, return, and midpoint strategy	100/10/75 item units
	iterated local search with tabu thresholding	100/10/75 item units
Koch and Wäscher (2016)	group-oriented genetic algorithm	60/25/75 item units
Hong and Kim (2017)	mathematical model based on the S-shape strategy	500/2.02/20 item units

Table 2.2: Exact and metaheuristic solution approaches for the standard order batching problem. For each reference, the solution approach and details on the size of the problem instances used are given. The problem instance size is given by three values, where the first value describes the maximum number of customer orders, the second value denotes the maximum number of items per customer order, and the third value represents the maximum picking device capacity specified by the number of customer orders, item units, or weight units.

address the standard OBP. Although the exact solution methods are able to cope with small-sized instances, they have shortcomings to consistently solve larger instances.

- Because the standard OBP is NP-hard if the number of customer orders per batch is greater than two, many studies focus on developing heuristic and meta-heuristic solution methods to solve the problem.
- Most of the solution approaches to the standard OBP have not been tested on problem instances of practically relevant size. Those that have been tested on larger instances do not exhibit an outstanding performance with respect to solution quality and/or the runtime required to solve the respective instances.
- It is surprising that an ALNS has not been proposed to address the standard OBP yet despite its convincing performance on combinatorial optimization problems related to the standard OBP.

2.5.4.3 Variants of order batching problems

The efficiency of the order picking process does not only depend on the grouping of customer orders into batches but also on the routing of order pickers through the warehouse, on how the batches are assigned to order pickers (in warehouses in which multiple order pickers operate), and on the sequence in which the batches are processed by the order pickers (Henn and Schmid 2013). Therefore, approaches which focus on the simultaneous solution of (some of) these planning problems are also considered in the OBP literature. Most variants of OBPs consider other objective functions, multiple end depots, and/or additional constraints such as precedence constraints. Although the joint consideration of order batching and other planning problems such as item storage assignment (see, e.g., Xiang et al. 2018) or zoning (see, e.g., Yu and de Koster 2009) further influences the efficiency of the order picking process, such publications are not discussed in the following because no decisions are made in this respect within this dissertation.

OBP integrating the sequencing of batches A variant of an OBP integrating the sequencing of batches is presented in Henn and Schmid (2013). The authors present an iterated local search and ABHC algorithm to solve the problem with the objective of minimizing the total tardiness for a set of customer orders.

OBPs integrating a PRP Variants of OBPs integrating a PRP are discussed in Kulak et al. (2012), Grosse et al. (2014), Matusiak et al. (2014), Cheng et al. (2015), Scholz and Wäscher (2017), Valle et al. (2017), and Valle and Beasley (2020). The problem considered in Kulak et al. (2012) aims at minimizing the total tour length for collecting the items of all batches in an arbitrary warehouse layout. To solve their OBP, a TS is proposed. Several heuristics, originally applied to the well-known traveling salesman problem, are used for addressing the PRP. Grosse et al. (2014) consider a joint OBP and PRP for a single-block parallel-aisle warehouse layout. Splitting of customer orders is allowed. The OBP is solved by an SA algorithm and the PRP by a savings heuristic as well as the return, midpoint, and largest gap picker routing strategy. Matusiak et al. (2014) consider an order picking system with a single-block parallel-aisle warehouse with three bidirectional cross aisles and multiple end depots and precedence-constrained picker routing. An SA algorithm is developed for their OBP and the exact A*-algorithm, proposed by Hart et al. (1968), is used to address precedence constraints in their PRP. Cheng et al. (2015) study a joint OBP and PRP for an arbitrary warehouse layout with the objective of minimizing total tour length. The authors propose a hybrid approach based on particle swarm optimization to solve their OBP and ant colony optimization to solve their PRP. Scholz and Wäscher (2017) investigate an OBP and PRP for a two-block parallel-aisle warehouse. The authors integrate several routing algorithms into an iterated local search approach for the OBP. Valle et al. (2017) study a joint OBP and PRP for a multi-block parallel-aisle warehouse layout with the objective of minimizing the total tour length. They propose a mathematical model formulation with several valid inequalities based on a sparse graph representation of the warehouse to prevent solutions that violate subtour breaking constraints. Briant et al. (2020) present an exponential linear programming formulation for a joint OBP and PRP for a multi-block parallel-aisle warehouse layout. Valle and Beasley (2020) propose a mathematical model which decides on the batching of customer orders with the objective of minimizing an approximation of the picker routing distance traveled. Once the order batching decision has been made, order picker routes are optimally determined.

OBPs integrating the sequencing of batches and the assignment of batches to order pickers Variants of OBPs integrating the sequencing of batches and the assignment of batches to order pickers are studied in Hong et al. (2012), Henn

(2015), and Matusiak et al. (2017). Hong et al. (2012) consider an integrated order batching, batch sequencing and batch assignment problem. Their problem assumes multiple order pickers operating in narrow picking aisles, in which congestion in the form of picker blocking may occur. The objective of their problem is to minimize the total retrieval time, which consists of travel time, pick time, and congestion delays caused by picker blocking. The authors develop a mixed integer programming approach, which is shown to be able to cope with small-sized instances and an SA-based approach for solving larger instances. Henn (2015) present an integrated order batching, batch assignment and sequencing problem, in which customer orders have to be grouped to batches, batches assigned to a limited number of order pickers, and batches scheduled such that the total tardiness is minimized. The problem is solved by means of a variable neighborhood descent and a variable neighborhood search approach. Matusiak et al. (2017) study an order batching, batch sequencing and batch assignment problem for an arbitrary warehouse layout with multiple end depots integrating individual differences in picking skills of order pickers. The objective of their problem is the minimization of total order pickers' batch execution times. The combined problem is solved by an ALNS.

OBPs integrating the sequencing of batches and a PRP Variants of OBPs integrating the sequencing of batches and a PRP are addressed in Won and Olafsson (2005), Tsai et al. (2008), and Chen et al. (2015). Won and Olafsson (2005) study an integrated order batching, batch sequencing, and PRP for an arbitrary warehouse layout. The objective is to minimize the total tour length and throughput time of all batches. The authors propose a simple construction heuristic, which first groups the customer orders into batches and then sequences the batches. A 2-opt heuristic is used to address the PRP. Tsai et al. (2008) investigate an OBP integrating batch sequencing and picker routing. In their problem, the total costs (depending on the total travel time) are minimized and both earliness and tardiness are penalized. Contrary to many problems described in the literature, splitting of customer orders is allowed. Thus, items of a single customer order can be collected on different order picking tours. To solve the problem, the authors propose a multiple genetic algorithm, which consists of a genetic algorithm for order batching and batch sequencing as well as a genetic algorithm for the PRP. Chen et al. (2015) propose a non-linear mixed integer programming approach to model the decisions on order batching, batch sequencing, and routing of a single order picker with the objective of

minimizing the total tardiness of all customer orders. A genetic algorithm is applied to the order batching and batch sequencing problem, and ant colony optimization is applied to solve their PRP. In numerical studies, problem instances with up to eight customer orders are considered.

OBP integrating the assignment of batches to order pickers, the sequencing of batches, and a PRP Finally, Scholz et al. (2017) introduce a mathematical model and a variable neighborhood descent algorithm for their order batching, batch assignment and sequencing problem integrating a PRP. Because the size of their model increases polynomially with the number of customer orders, small-sized instances can be solved to optimality within a reasonable amount of runtime.

Table 2.3 and Table 2.4 summarize the OBPs described above. In both tables, the references are sorted in ascending order of the year of publication and alphabetically by the surname of the first (and second) author. The first column of both tables includes the respective reference. Table 2.3 outlines OBPs integrating the sequencing of batches, the assignment of batches to order pickers, and/or a PRP. Columns two to four of Table 2.3 indicate the planning problems considered by the respective reference. The fifth column specifies the objective function, and the sixth column details further problem characteristics. Whenever a reference integrates the sequencing of batches, the assignment of batches to order pickers, and/or the PRP into the OBP, this is indicated by a bullet point in the corresponding cell of the table. Table 2.4 provides an overview of the solution approaches to OBPs integrating the sequencing of batches, the assignment of batches to order pickers, and/or a PRP. Columns two to five specify the solution approach(es) for the planning problem considered by the respective reference.

The main findings of our literature review on OBP variants are as follows:

- Variants of OBPs primarily differ with respect to the warehouse layout (e.g., number of blocks and width of picking aisles), the underlying objective function (e.g., tardiness-related), and characteristics of the order picking process (e.g. consideration of individual order picker skills, precedence-constrained order picking, or order picker blocking).
- Although in many real-world applications customers expect to receive their requested items within a certain time period, customer order due dates and due date-related objectives are only studied in a few papers.

Reference	BS	BA	PR	Objective(s)	Further problem characteristic(s)
Won and Olafsson (2005)	•		•	minimization of total tour length and throughput time	arbitrary warehouse layout
Tsai et al. (2008)	•		•	minimization of total travel costs and penalization of earliness and tardiness	arbitrary warehouse layout, splitting of customer orders is allowed
Hong et al. (2012)	•	•		minimization of total retrieval time	single-block parallel-aisle warehouse with narrow picking aisles, multiple order pickers, picker blocking
Kulak et al. (2012)			•	minimization of total tour length	arbitrary warehouse layout
Henn and Schmid (2013)	•			minimization of total tardiness	single-block parallel-aisle warehouse
Grosse et al. (2014)			•	minimization of total tour length	single-block parallel-aisle warehouse, splitting of customer orders is allowed
Matusiak et al. (2014)			•	minimization of total tour length	single-block parallel-aisle warehouse with three bidirectional cross aisles, precedence constraints, multiple end depots
Chen et al. (2015)	•		•	minimization of total tardiness	arbitrary warehouse layout
Cheng et al. (2015)			•	minimization of total tour length	arbitrary warehouse layout
Henn (2015)	•	•		minimization of total tardiness	arbitrary warehouse layout
Matusiak et al. (2017)	•	•		minimization of total order pickers' batch execution times	arbitrary warehouse layout, multiple end depots, individual order picker skills
Scholz et al. (2017)	•	•	•	minimization of total tardiness	arbitrary warehouse layout
Scholz and Wäscher (2017)			•	minimization of total tour length	two-block parallel-aisle warehouse
Valle et al. (2017)			•	minimization of total tour length	multi-block parallel-aisle warehouse
Briant et al. (2020)			•	minimization of total tour length	multi-block parallel-aisle warehouse
Valle and Beasley (2020)			•	minimization of an approximation of total tour length	multi-block parallel-aisle warehouse

Table 2.3: Overview of order batching problems integrating the sequencing of batches (BS), the assignment of batches to order pickers (BA), and/or picker routing (PR).

Reference	Order batching	Batch sequencing	Batch assignment	Picker routing
Won and Olafsson (2005)	construction heuristic	construction heuristic		2-opt heuristic
Tsai et al. (2008)	genetic algorithm	genetic algorithm		genetic algorithm
Hong et al. (2012)	mathematical model and simulated annealing	mathematical model and simulated annealing	mathematical model and simulated annealing	
Kulak et al. (2012)	tabu search			several traveling salesman problem algorithms
Henn and Schmid (2013)	attribute-based hill climber and iterated local search	attribute-based hill climber and iterated local search		
Grosse et al. (2014)	simulated annealing			savings algorithm and return, midpoint and largest gap strategy
Matusiak et al. (2014)	simulated annealing			exact A*-algorithm proposed by Hart et al. (1968)
Chen et al. (2015)	genetic algorithm	genetic algorithm		ant colony optimization
Cheng et al. (2015)	particle swarm optimization			ant colony optimization
Henn (2015)	variable neighborhood descent and variable neighborhood search	variable neighborhood descent and variable neighborhood search	variable neighborhood descent and variable neighborhood search	
Matusiak et al. (2017)	adaptive large neighborhood search	adaptive large neighborhood search	adaptive large neighborhood search	
Scholz et al. (2017)	mathematical model and variable neighborhood descent	mathematical model and variable neighborhood descent	mathematical model and variable neighborhood descent	mathematical model and variable neighborhood descent
Scholz and Wäscher (2017)	iterated local search			several routing algorithms
Valle et al. (2017)	mathematical model			mathematical model
Briant et al. (2020)	mathematical model			mathematical model
Valle and Beasley (2020)	mathematical model			mathematical model

Table 2.4: Overview of solution approaches for order batching problems integrating the sequencing of batches, the assignment of batches to order pickers, and/or a picker routing problem.

- Recently, research focuses on the simultaneous consideration of order batching and other planning problems. As order batching simultaneously arises with the routing of an order picker, their joint consideration has received the most attention so far.
- Solution approaches for integrated problems are in general based on a single metaheuristic paradigm to address the OBP (and batch sequencing and/or batch assignment) and on an exact or construction algorithm for routing an order picker.

2.5.5 Zone picking

In zone picking (or zoning), the picking area of a warehouse is divided into (smaller) picking areas, called zones, each with one or few order pickers assigned to it (Yu and de Koster 2009). In a zone picking system, each order picker is responsible for retrieving the items of a picking order that are stored in her zone.

Depending on whether the items of a picking order are collected successively or simultaneously in the zones, progressive zoning and parallel zoning can be distinguished (de Koster et al. 2012). With progressive zoning, also called pick-and-pass zoning, the items of a customer order are sequentially picked zone by zone as described in the following: An order picker retrieves the items of a picking order stored in her zone. Once she has collected all the items from her zone, she places the items into a bin, which is transferred by a conveyor to the next order picker, who collects the items of the picking order stored in her zone. A picking order is completed after having collected all the requested items from the relevant zones and after having transferred them to the depot, where they are prepared for shipment. In parallel zoning, also called synchronized zoning, the requested items of a picking order are simultaneously retrieved by multiple order pickers in multiple zones. Consequently, the items need to be consolidated before they can be packed and shipped to a customer.

Advantages of zone picking include a reduction of an order picker’s travel time because she does not travel between the zones or between the depot and the picking area. Moreover, an order picker’s travel time can be reduced because of a reduction of congestion within a picking aisle. If items are not randomly assigned to the storage locations in the zones, performance gains with respect to the time to search for and to identify the items can be achieved because of an increase in familiarity with the

item storage locations (de Koster et al. 2012). Although progressive zone picking may result in a low pick rate (items picked per time unit), it does not require that customer orders have to be consolidated after the order picking process. Contrary to this, in parallel zone picking, an order picker’s pick rate may be high, but the consolidation of customer orders before shipment is required (Parikh and Meller 2008).

Compared to other planning problems in order picking, zone picking has been only rarely considered in the literature (de Koster 2015, van der Gaast 2016). For a generic discussion on zoning, we refer the reader to Speaker (1975). Gray et al. (1992) propose a hierarchical approach of mathematical models to evaluate the economic trade-offs of equipment and technology selection, item storage assignment, number of zones, picker routing, and order batching when designing a zoning system. Malmberg (1996) investigates the trade-offs in space requirements and order picking efficiency in a zoning system with dedicated and randomized storage. Jane (2000) proposes a heuristic method for assigning items to storage zones with the objective of balancing workloads among all order pickers. Petersen (2002) examines the effect of single customer order picking, order batching, and different item storage assignment and picker routing strategies on the traveling of order pickers within a simulation study. The author shows that the number of picking aisles per zone, the length of picking aisles, the number of items contained in a picking order, and the item storage assignment strategy significantly influence the average travel distance of an order picker within a zone. Ho and Chien (2006) compare two sequencing strategies according to which an order picker visits zones in a distribution center. Different technology requirements and item storage assignment strategies are examined in Eisenstein (2008). Parikh and Meller (2008) investigate the problem of selecting between a batch picking and a zone picking strategy. A cost model is proposed to estimate the cost for both picking strategies. Yu and de Koster (2008) model a progressive zoning system as a Jackson queuing network for estimating throughput times of customer orders and average work in process. The resulting estimates can be used for determining the number of zones. Pan and Wu (2009) develop an analytical model for a progressive zoning system in which the operation of an order picker is described as a Markov Chain to estimate the expected travel distance of an order picker. Based on the model, three exact algorithms are proposed to assign items to storage locations for the cases of a warehouse with a single picking zone, a warehouse with unequal-sized zones, and a warehouse with

equal-sized zones. Yu and de Koster (2009) propose an approximation model based on queuing network theory to analyze the impact of order batching and picking area zoning on the average throughput time of customer orders in a synchronized order picking system. Melacini et al. (2011) model a progressive zoning system by a network of queues to estimate the mean order throughput and use analytical models to estimate the travel distance. De Koster et al. (2012) formulate a mathematical model to determine the optimal number of zones in a parallel zoning system such that the total order picking time is minimized. A comprehensive study of zone picking systems considering single-segment and multi-segment routing, and congestion and blocking at conveyors is provided in van der Gaast (2016). Moreover, different item storage assignment strategies are compared for the case of a real-world zoning system.

2.5.6 AGV-assisted order picking

Another alternative to reduce the travel times of order pickers in picker-to-parts systems without extensive organizational overhead is to transport the retrieved items (e.g., in bins, on pallets, in containers, or in roll cages if large-sized items are collected) using AGVs (Boysen et al. 2019a, de Koster 2018, Azadeh et al. 2019). An AGV is a computer-controlled and wheel-based load carrier, which is automatically guided by a combination of software and sensor-based system along a prescribed path (Material Handling Institute 2020). Figure 2.8 depicts an AGV handling multiple customer order bins.

AGVs can support an order picking process as follows: For instance, the AGV-PickTM developed by Swisslog, or the Pick-n-GoTM developed by Kollmorgen, automatically follows an order picker closely through the warehouse so that the order picker can place the retrieved items onto the AGV. Once all items of a picking order have been collected, the AGV autonomously transports the items to the depot. Here, the retrieved items are prepared for shipment to the respective customers, and the AGV is equipped (e.g., with empty bins) for collecting the items of the next picking order. The order picker remains in the picking area, and if available, another AGV is requested in time to meet the order picker at the first picking location defined by the successive picking order.

In another variant developed by Locus Robotics, the so-called LocusBotsTM does not accompany an order picker. Instead, the AGV autonomously drives to a picking



Figure 2.8: SSI Schäfer's FTS 2PICK™. The figure depicts SSI Schäfer's FTS 2PICK™ handling multiple customer order bins (SSI Schäfer 2016).

location and waits there for an order picker to load the requested item, which the AGV announces on its display. Once an order picker has placed the requested item onto the AGV, the AGV proceeds to the next picking location, at which an order picker has to execute the necessary pick (see, e.g., Azadeh et al. 2019, Boysen et al. 2019a). When the picking order is complete, the AGV returns to the depot. A variant of this kind of AGV-supported order picking system is considered in Chapter 5.

The concept of AGV-assisted order picking has the following advantages: Compared to the order picking process described in Section 2.3.2, AGV-assistance allows order pickers to continuously retrieve items because they remain in the picking area without returning to the depot each time a picking order is completed or the capacity of the picking device is exhausted. Thus, the throughput of completed customer orders can be increased. Moreover, integrating AGVs into an existing picker-to-part system hardly changes the basic order fulfillment process (Löffler et al. 2020). Therefore, AGV-assisted order picking can easily be implemented without extensively redesigning the system or installing further technologies. Compared to other concepts based on fixed hardware (e.g., conveyors or automated storage and retrieval systems), AGV-assisted order picking enables a quicker adaption to varying

workloads by temporarily adding (or removing) AGVs, e.g., when the number of customer orders increases due to season sales.

Disadvantages of AGV-assisted order picking systems concern the investment costs and regular operating costs for the AGV fleet. However, Löffler et al. (2020) state that the additional investments for AGVs are quickly amortized by the reduction of unnecessary trips from the picking area back to the depot. A challenge of AGV-assisted order picking concerns the synchronization of human order pickers and AGVs, which makes the modeling, analysis and optimization of AGV-assisted order picking systems completely different from traditional picker-to-parts systems (Azadeh et al. 2019). It is one of the main intentions of this dissertation to address this challenge.

In the literature, AGV-assisted order picking has only been studied by Löffler et al. (2020) yet. In the considered setting, an order picker is closely accompanied by an AGV during the order picking process as described above. The authors address the routing of AGV-assisted order pickers with both a given and a facultative processing sequence of incoming customer orders. The objective of their problems is the minimization of the travel length for collecting the items of a customer order. To solve the problem with given processing sequence of customer orders, the algorithm of Ratliff and Rosenthal (1983) is extended and repeatedly solved as an integrated part of a dynamic programming approach. To address the problem with facultative processing sequence of customer orders and to evaluate alternative sequences, this procedure is integrated into several heuristic approaches. Löffler et al. (2020) show that compared to an order picking system without the support of AGVs, in which an order picker returns to the depot after each order picking tour, a reduction of travel distance of approximately 20% can be achieved by AGV-assisted order picking.

2.6 Metaheuristics used in this dissertation

2.6.1 Classification of metaheuristics

Metaheuristics are solution methods that use higher-level strategies that guide underlying heuristics to explore a solution space efficiently and effectively (Talbi 2009). Contrary to exact solution methods, metaheuristics do not guarantee to find an optimal solution to the problem considered (Sörensen 2015). Rather, they aim to achieve a good trade-off between the solution quality and the computational effort

required to solve the problem (see, e.g., Talbi 2009).

Metaheuristics can be classified according to the number of solutions considered at any time of the search process into single solution (or also called trajectory methods) and population-based methods (see, e.g., Blum and Roli 2003, Gendreau and Potvin 2005, Talbi 2009). Single solution metaheuristics are algorithms considering a single solution at any time during the search process. Examples of single solution methods are methods based on local search like large neighborhood search (LNS), ALNS, SA, and TS. Population-based metaheuristics, on the contrary, work on a set of solutions at any time during the search process. Examples of population-based methods are evolutionary computation and ant colony optimization. All solution methods developed in this dissertation are based on single solution metaheuristics.

2.6.2 Tabu search

TS is a metaheuristic based on local search that was originally proposed by Glover (1986). It has been applied to various combinatorial optimization problems providing near-optimal solutions in moderate runtimes. For a detailed description of TS and its extensions, we refer to Gendreau and Potvin (2019). In Figure 2.9, we give a pseudocode overview of a TS.

```

 $s \leftarrow \text{generateInitialSolution}()$ 
 $\text{tabuList} \leftarrow \emptyset$ 
repeat
   $s \leftarrow \text{chooseBestOf}(\mathcal{N}(s) \setminus \text{tabuList})$ 
   $\text{update}(\text{tabuList})$ 
until stop criterion is met

```

Figure 2.9: TS in pseudocode.

At each iteration, TS examines the neighborhood $\mathcal{N}(s)$ of a current solution s . The current solution s is modified by a set of neighborhood operators, called moves. To escape from low-quality local optima, TS selects the best neighbor of the current solution as the new current solution, even if the best neighbor is of lower quality than the current solution (Blum and Roli 2003). To avoid short-term cycling of the search process, moves towards recently visited solutions are prohibited, and these solutions or attributes of these solutions are inserted and are stored in a tabu list for a given number of iterations, called tabu tenure. Thus, the neighborhood of the current solution contains only those solutions that are not stored in the tabu list.

After the new current solution has been added to the tabu list, one of the solutions that were already in the tabu list is removed, usually in a first-in first-out order (Blum and Roli 2003).

The tabu tenure controls the memory of the search process and has a significant impact on the performance of a TS (Talbi 2009). In the case of small tabu tenures, the search process will focus on small areas of the solution space, and the probability of cycling increases. In the case of large tabu tenures, larger parts of the solution space are explored because revisiting a higher number of solutions is forbidden.

The tabu tenure can be static or dynamic. In the static form, a static value is given for the tabu tenure. In the dynamic form, the tabu tenure is varied during the search process (Blum and Roli 2003). For example, Battiti and Tecchiolli (1994) implement a dynamic tabu tenure, which is increased if solutions are repetitive (thus a higher diversification is required). If the objective function value is not improved (thus intensification is required), the tabu tenure is decreased.

Another way to explicitly use historical information about the search process are medium-term and long-term memories (Talbi 2009). The medium-term memory stores information about the elite (e.g., best) solutions found during the search process to guide the search process in promising regions of the solution space. The idea is to extract the (common) features of the elite solutions and then to intensify the search process around solutions providing those characteristics (Talbi 2009). Thus, priority is given to attributes of the set of elite solutions. A possible approach is to restart the search process with the best solution and then to fix in this solution the most promising components extracted from the elite solutions (Talbi 2009).

In the long-term memory, information about the visited solutions during the search process is stored (Talbi 2009). The idea is to encourage the search process towards unvisited areas of the solution space. For example, to diversify the search, attributes of elite solutions can be discouraged.

Whether intensification or diversification of the search process is useful, depends on the landscape of the underlying optimization problem. For instance, if promising solutions are located distant to each other in the search space, intensifying the search will probably not guide the search towards high-quality solutions (Talbi 2009).

Because a tabu list may be sometimes too restrictive (Talbi 2009), e.g., tabus may lead to a stagnation of the search process, Glover (1989) introduce aspiration criteria which can override the tabu status of a move. For instance, an aspiration

criterion could select a tabu move if a better solution than the current best known solution is found.

Several stop criteria are possible, but typically the algorithm stops after a given number of iterations, a fixed amount of computational time, if the best found solution has not improved for a given number of iterations, as soon as the objective function value reaches a given threshold value, or if all solutions in the neighborhood $\mathcal{N}(s)$ are forbidden by the tabu list.

2.6.3 Large neighborhood search

LNS was introduced by Shaw (1997) and belongs to the class of very large neighborhood search algorithms as presented in Ahuja et al. (2002). The idea of these algorithms is that the exploration of a large neighborhood can lead to local optima of high quality. Thus, a more promising search path can be followed, and better solutions can be found (Pisinger and Ropke 2019). However, exploring a large neighborhood is time-consuming. For this reason, filtering techniques are applied, which aim at focusing the search on an optimal or a near-optimal neighboring solution.

In LNS, the neighborhood is implicitly defined by a destroy and a repair operator. A destroy operator destructs a part of the current solution, and a repair operator rebuilds the destroyed solution (Pisinger and Ropke 2019). Schrimpf et al. (2000) proposed a similar approach as ruin and recreate. An overview of a pseudocode for a LNS is given in Figure 2.10. In the pseudocode, we use the following notation: s denotes the current solution, s' is a temporary solution that can be rejected or defined as the current solution, and s^* represents the best found solution during the search process. Function $h^-(\cdot)$ describes the destroy operator, and function $h^+(\cdot)$ specifies the repair operator. The objective function is denoted by $f(\cdot)$.

Starting with the current solution s , LNS first applies the destroy operator $h^-(s)$ to destroy a part of the current solution s . Then, the repair operator $h^+(s)$ rebuilds the destroyed solution. Subsequently, LNS returns a new feasible solution s' . In the next step, the acceptance function $accept(s')$ is used to decide whether s' becomes the current solution or not. Afterwards, we check whether s' is better than the best known solution s^* . If $f(s') < f(s^*)$, the best known solution s^* is replaced by s' . The described procedure is repeated until the stop criterion is satisfied. Then, the best found solution during the search process is returned.

Several acceptance and stop criteria are possible: An acceptance criterion could

```

 $s \leftarrow \text{generateInitialSolution}()$ 
 $s^* \leftarrow s$ 
repeat
   $s' \leftarrow h^+(h^-(s))$ 
  if  $\text{accept}(s')$  then
     $s \leftarrow s'$ 
  end if
  if  $f(s') < f(s^*)$  then
     $s^* \leftarrow s'$ 
  end if
until stop criterion is met
return  $s^*$ 

```

Figure 2.10: LNS in pseudocode.

be to accept only improving solutions. Another acceptance criterion is based on SA. Here, the temporary solution s' is always accepted if $f(s') \leq f(s)$. In the case of $f(s') > f(s)$, s' is accepted with the Metropolis probability $e^{-(f(s')-f(s))/t_i}$, where $t_i > 0$ is a temperature parameter (Metropolis et al. 1953). A solution s' is more likely to be accepted if the temperature is high and the increase in the objective function value is low. The starting temperature $t_0 > 0$ is set to a relatively high value to allow deteriorating solutions to be accepted at the beginning of the search process. As the search process progresses, the temperature is gradually decreased according to a predefined cooling schedule. Thus, the probability of allowing deteriorating solutions to be accepted decreases, and the algorithm stops in a local optimum (Pisinger and Ropke 2019). Typical stop criteria are defined by a fixed number of iterations or by a fixed computational time.

When designing a destroy operator, it should be considered that the entire solution space or at least the promising part of the solution space, in which the global optimum is expected, can be reached. Typically, a destroy operator contains a stochastic component to destroy different parts of the current solution. Besides, the degree of destruction is a crucial issue for the performance of the LNS: On the one hand, if the degree of destruction is relatively small (with respect to the problem size), only a small part of the solution is destroyed. On the other hand, the higher the degree of destruction is, the more the problem is resolved from scratch, and the more computational time is required (Ropke and Pisinger 2006a). To control the size of the neighborhood, Shaw (1998) propose to gradually increase the degree of destruction. Ropke and Pisinger (2006a) randomly choose the degree of destruction

at each iteration from a given interval depending on the instance size.

When designing a repair operator, it has to be decided whether an exact or a heuristic solution method should be used. An exact repair operator can lead to high-quality solutions but often at the expense of a longer computational time. Moreover, if only a small part of the solution is destroyed in each iteration, applying an exact repair operator may be disadvantageous from a diversification point of view: Because exact solution methods guarantee to find optimal solutions, only improving solutions or solutions with identical objective function value are generated. Thus, the exploration of the solution space is rather limited to find local optima (Pisinger and Ropke 2019). Even if heuristic repair operators seem to generate less promising solutions at first, they may diversify the search and therefore may allow to reach promising areas of the solution space (Ropke and Pisinger 2006a).

2.6.4 Adaptive large neighborhood search

ALNS, originally proposed by Ropke and Pisinger (2006a), is an extension of LNS. Because it may depend on the specific instance which destroy (or repair) operator is appropriate to apply at a certain time, ALNS uses multiple destroy and repair operators throughout the search process. An advantage of alternating between different destroy and repair operators is that the algorithm can be guided towards a more robust search of the solution space. Because repair operators often tend to perform moves that seem to be locally best, Ropke and Pisinger (2006a) propose to add a noise term to the objective function to randomize the repair operators. Moreover, an adaptive mechanism for the selection of these operators is used. At each iteration, a destroy and a repair operator are chosen based on a probability distribution that depends on the performance of the destroy and repair operators in past iterations. An overview of a pseudocode for an ALNS is given in Figure 2.11.

In the pseudocode, we use the following notation: s denotes the current solution, s' is a temporary solution that can be rejected or defined as the current solution, and s^* represents the best solution found during the search process. The objective function is defined by $f(\cdot)$. Compared to the pseudocode of a LNS, in ALNS, several destroy and repair operators are considered and are defined by the functions $h_i^-(\cdot)$ and $h_i^+(\cdot)$, where $i \in \mathcal{X}$ denotes the respective operator. Moreover, ALNS uses variables π_i^- and π_i^+ , which describe the probability value for selecting a destroy and repair operator $i \in \mathcal{X}$. To avoid repetition, we describe only those components of

```

 $s \leftarrow \text{generateInitialSolution}()$ 
 $s^* \leftarrow s$ 
repeat
  choose destroy and repair operators  $(h_i^-, h_i^+)$  according to probabilities  $(\pi_i^-, \pi_i^+)$ 
   $s' \leftarrow h_i^+(h_i^-(s))$ 
  if  $\text{accept}(s')$  then
     $s \leftarrow s'$ 
  end if
  if  $f(s') < f(s^*)$  then
     $s^* \leftarrow s'$ 
  end if
  adjust the weights  $w_i$  and probabilities  $\pi_i$  of the destroy and repair operators
until stop criterion is met
return  $s^*$ 

```

Figure 2.11: ALNS in pseudocode.

the pseudocode that differ from the pseudocode of the LNS presented above.

An adaptive weight adjustment procedure dynamically evaluates the importance of each operator during the search process. In this context, the selection probability π_i of operator $i \in \mathcal{X}$ is modified based on the performance of the operator in previous iterations according to a roulette wheel selection procedure as proposed in Ropke and Pisinger (2006a). At each iteration, the selection probability of operator $i \in \mathcal{X}$ is calculated as $\pi_i = w_i / \sum_{i \in \mathcal{X}} w_i$, where w_i corresponds to the weight of operator $i \in \mathcal{X}$. Initially, all operators are assigned the same weight ω .

The performance of an operator $i \in \mathcal{X}$ is measured in terms of a scoring system. Let o_i denote the score value of operator $i \in \mathcal{X}$. If a previous destroy-repair operation resulted in a new global best solution, the current scores of the respective operators are increased by o_{best} , if a previous destroy-repair operation led to a better solution that has not been found before by o_{imp} , and if a new deteriorating solution is found but accepted according to a given acceptance criterion by o_{acc} . Note that the scores for both operators are increased by the same value because the destroy-repair operation to be rewarded cannot be clearly attributed to one of the two operators.

The search process of an ALNS is divided into a number of segments of γ iterations. After γ iterations, weights are updated and the new weight w_{i+1} is determined as $w_{i+1} = (1 - r) \cdot w_i + r \cdot \frac{o_i}{\beta_i}$. The parameter $r \in [0, 1]$ controls the reaction speed of the weight adjustment and takes the success of an operator in previous segments into account. For example, if r is zero, the initial weight values are maintained. The number of times that an operator $i \in \mathcal{X}$ was chosen in the previous segment is

denoted by β_i . The values of o_i and β_i are set to zero after each adjustment of the weights.

With respect to possible acceptance and stop criteria, we refer to those described above (see LNS).

Chapter 3

A metaheuristic hybrid of adaptive large neighborhood search and tabu search for the standard order batching problem

The contents of this chapter are included in similar form in the following publication: I. Žulj, S. Kramer, and M. Schneider. A hybrid of adaptive large neighborhood search and tabu search for the order batching problem. *European Journal of Operational Research*, 264(2):653–664, 2018.

Among all order picking activities in picker-to-parts systems (see Section 2.4), it is estimated that picker traveling is the most time-consuming activity with a share of 50% and more of total order picking time. Consequently, reducing unproductive picker traveling time is critical for any order picking system that relies on human order pickers. Travel time (or travel distance) mainly depends on the assignment of items to storage locations, the grouping of customer orders into batches, and the routing of order pickers through the warehouse. This chapter focuses on order batching, which has a strong influence on the efficiency of warehouse operations.

The remainder of this chapter is structured as follows. In Section 3.1, we provide a mathematical description of the standard OBP. To solve the problem, we develop a metaheuristic hybrid that combines an ALNS with a TS (see Section 3.2). The performance of ALNS×TS is investigated in extensive numerical studies in Section 3.3. To this end, we first perform tests on a set of newly generated large-scale instances

with up to 600 customer orders. Subsequently, we assess the performance of the hybridization in comparison to ALNS and TS as standalone methods. Finally, we conduct an extensive comparison of ALNS×TS to all previously published OBP methods that have been tested on (any subset of) the standard OBP instances from the literature to investigate their performance. In Section 3.4, we conclude this chapter with a summary.

3.1 Problem description

In this section, we describe the integer programming model for the standard OBP, originally proposed by Gademann and van de Velde (2005). In their model formulation, the set of all feasible batches I not exceeding the capacity of the picking device is determined before solving an OBP instance. The picking device capacity is indicated by parameter C and is expressed in item units. Let $J = \{1, \dots, n\}$ denote the set of customer orders, and let parameters c_j represent the capacity needed for customer order $j \in J$. It is assumed that $c_j \leq C$ for all $j \in J$ to guarantee the feasibility of the problem. The vector $a_i = (a_{i1}, \dots, a_{in})$ represents feasible batches, where $a_{ij} = 1$ if customer order $j \in J$ is included in batch $i \in I$, otherwise $a_{ij} = 0$. To ensure that only feasible batches are considered, we define constraints (3.1) as follows:

$$\sum_{j \in J} c_j \cdot a_{ij} \leq C \quad \forall i \in I \quad (3.1)$$

The tour length for collecting the items of batch $i \in I$ is denoted by parameters l_i and is determined according to the underlying picker routing strategy (e.g., S-shape). Binary decision variables x_i indicate whether batch i is chosen from the set of all feasible batches I ($x_i = 1$) or not ($x_i = 0$). A batch $i \in I$ to which $x_i = 1$ applies is referred to as selected batch in the following. Then, the standard OBP can be formulated as follows:

$$\text{minimize } \sum_{i \in I} l_i \cdot x_i \quad (3.2)$$

subject to

$$\sum_{i \in I} a_{ij} \cdot x_i = 1 \quad \forall j \in J \quad (3.3)$$

$$x_i \in \{0, 1\} \quad \forall i \in I \quad (3.4)$$

Objective function (3.2) minimizes the total tour length for collecting the items of all selected batches. Constraints (3.3) assign each customer order to exactly one of the selected batches. In constraints (3.4), we define the decision variables as binary.

3.2 Solution method

In this section, we describe our hybrid solution approach of ALNS and TS to address the standard OBP. In Figure 3.1, we give a pseudocode overview of ALNS \times TS. First, we generate an initial solution using the C&W(ii) algorithm (see Section 3.2.1). Subsequently, the solution is improved by several ALNS iterations, in which neighboring solutions are accepted according to an SA-based decision criterion (see Section 3.2.2). A TS is run (Section 3.2.3) after a certain number of ALNS iterations.

```

generate initial solution  $s$  using the C&W(ii) algorithm
 $s^* \leftarrow s$ 
repeat
    randomly draw  $q^-$  customer orders to be removed
    choose destroy and repair operators  $(h_i^-, h_i^+)$  according to probabilities  $(\pi_i^-, \pi_i^+)$ 
     $s' \leftarrow h_i^+(h_i^-(s))$ 
    if certain number of ALNS iterations has passed then
         $s' \leftarrow TS(s)$ 
    end if
    if acceptSA( $s'$ ) then
         $s \leftarrow s'$ 
    end if
    if  $f(s') < f(s^*)$  then
         $s^* \leftarrow s'$ 
    end if
    adjust the weights  $w_i$  and probabilities  $\pi_i$  of the destroy and repair operators
until stop criterion is met
return  $s^*$ 

```

Figure 3.1: Overview of the ALNS \times TS algorithm.

3.2.1 Initial solution

To generate an initial solution for the standard OBP, we use the C&W(ii) algorithm as described in Section 2.5.4. In preliminary tests, we compared the solution quality of ALNS \times TS using C&W(i) and C&W(ii) on a benchmark set that is described in Section 3.3.2. Table 3.1 presents an aggregate overview of the test results and

reports averages for groups of instances defined by the number of customer orders (column # orders) and the carrying capacity of the picking device (column C). For ALNS×TS using C&W(i) and ALNS×TS using C&W(ii), we give the following information:

- f_s denotes the initial solution as the average of the objective function values obtained by C&W(i) or C&W(ii) over each of the individual instances.
- f_e indicates the final solution as the average of the best objective function values obtained by ALNS×TS using C&W(i) or ALNS×TS using C&W(ii) over each of the individual instances.
- Δ_f (%) reports the percentage gap between f_s and f_e . The percentage gap is computed as $\Delta_f = 100 \cdot (f_s - f_e) / f_s$.

The experiment shows that there is a significant difference with respect to solution quality of the constructed solution, however, the final solution of ALNS×TS does not differ much: while C&W(ii) generates initial solutions that are by 1.3% better than those of C&W(i), the final solution is approximately the same (difference of about 0.0027%).

3.2.2 Adaptive large neighborhood search component

The motivation for designing an ALNS for the standard OBP is the convincing performance of this method on related combinatorial optimization problems like vehicle routing and scheduling problems. Furthermore, ALNS is known to provide robust solution quality for problem instances with different characteristics due to the integrated adaptive weight adjustment.

Our ALNS uses three destroy and two repair operators within the same search process (see paragraph *Destroy and repair operators*). Moreover, we apply an adaptive mechanism for the selection of these operators (see paragraph *Adaptive mechanism*). At each iteration, a destroy and a repair operator are chosen based on a probability distribution that depends on the performance of the destroy and repair operators in past iterations. To overcome local optima, we implement an SA-based acceptance criterion and add a noise term to the objective function (see paragraph *Acceptance criterion*).

# orders	C	ALNS×TS using C&W(i)			ALNS×TS using C&W(ii)		
		f_s	f_e	Δ_f (%)	f_s	f_e	Δ_f (%)
40	30	10844	10465	3.6	10799	10465	3.1
40	45	7396	6864	7.8	7284	6864	5.8
40	60	5718	5278	8.3	5601	5278	5.8
40	75	4645	4273	8.7	4518	4273	5.4
60	30	16003	15493	3.3	15964	15493	2.9
60	45	10684	10032	6.5	10578	10032	5.2
60	60	8340	7705	8.2	8179	7705	5.8
60	75	6824	6294	8.4	6627	6294	5.0
80	30	21334	20671	3.2	21324	20671	3.1
80	45	14155	13328	6.2	13990	13328	4.7
80	60	10960	10173	7.7	10734	10173	5.2
80	75	9007	8233	9.4	8672	8233	5.1
100	30	26370	25578	3.1	26370	25578	3.0
100	45	17369	16357	6.2	17116	16357	4.4
100	60	13476	12472	8.0	13086	12472	4.7
100	75	11041	10151	8.8	10690	10151	5.0
Average		12135	11460	6.7	11971	11460	4.6

Table 3.1: Comparison of ALNS×TS results using C&W(i) and C&W(ii) on the benchmark set UDD/S-shape. The first two columns specify the respective groups of instances defined by the number of customer orders (column # orders) and the carrying capacity of the picking device (column C). For ALNS×TS using C&W(i) and ALNS×TS using C&W(ii), we report the initial solution as the average of the objective function values obtained by C&W(i) or C&W(ii) over each of the individual instances (column f_s). We also report the final solution as the average of the best objective function values obtained by ALNS×TS using C&W(i) or ALNS×TS using C&W(ii) over each of the individual instances (column f_e). The percentage gap between f_s and f_e is given in columns Δ_f (%).

Destroy and repair operators Our ALNS uses the following destroy operators.

Random removal randomly removes q^- customer orders from the current solution in order to diversify the search.

Worst removal was introduced by Ropke and Pisinger (2006b) and removes customer orders that are unfavorably assigned to batches in the current solution in order to intensify the search. A number of q^- customer orders with long travel length in the current solution is removed. First, we compute for each customer order $f(j, s)$, which describes the change in total travel length if customer order j is removed from the current solution s . Then, we sort all customer orders in a list in descending order of $f(j, s)$. Let L denote the size of this list. We choose

the customer order at position $\lfloor L \cdot u^p \rfloor$ in this list, where u denotes a uniformly distributed number in $[0, 1]$ and p a randomization parameter in order to avoid repeatedly removing the same customer orders.

Shaw removal proposed by Shaw (1997) removes customer orders that are similar to each other according to several criteria. First, we define the relatedness between customer order i and j according to the tour length saving that may result when picking both customer orders simultaneously on a single order picking tour instead of two separate order picking tours. Then, we compute the normalized savings sav_{ij}^{norm} as

$$sav_{ij}^{norm} = \frac{d_i + d_j - d_{ij}}{\min(d_i, d_j)}, \quad (3.5)$$

where d_i and d_j denote the tour length for picking customer order i and customer order j on separate order picking tours, and d_{ij} represents the tour length when picking both customer orders simultaneously on a single order picking tour.

Furthermore, the relatedness is measured by considering the current assignment structure of customer orders to batches. Customer orders that are assigned to the same batch are likely to be exchangeable and thus considered more related to each other. We define binary coefficients b_{ij} to indicate whether customer order i and customer order j are assigned to the same batch ($b_{ij} = 1$) or not ($b_{ij} = 0$). Then, the relatedness measure r_{ij} between customer order i and customer order j is computed as follows (lower values correspond to more related customer orders):

$$r_{ij} = \frac{1}{sav_{ij}^{norm} + b_{ij}} \quad (3.6)$$

The first customer order i to be removed is randomly chosen. Non-removed customer orders j are sorted in a list in descending order of relatedness measure r_{ij} . We choose the customer order at position $\lfloor L \cdot u^p \rfloor$ in a list of size L , where again u denotes a uniformly distributed number in $[0, 1]$ and p a randomization parameter.

Our ALNS uses the following repair operators.

Greedy insertion iteratively assigns customer orders to the batch with minimal travel length increase.

Regret insertion was proposed by Ropke and Pisinger (2006b) and improves the greedy insertion by anticipating the future effect of an insertion operation. The k -regret value describes the change in total travel length of inserting customer order j in its best and its k -best batch. We implement the 2-regret and 3-regret heuristic.

Adaptive mechanism The adaptive weight adjustment procedure described in Section 2.6.4 is used to evaluate the importance of each operator during the search process. The adaptive mechanism modifies the probability with which an operator is chosen based on the performance of the operator in past iterations. Initially, all operators are set to the same weight ω . We use score value o_i to measure the performance of an operator $i \in \mathcal{X}$. If an operator finds a new best solution, the score is increased by o_{best} , if a better solution is found by o_{imp} , and if a new deteriorating solution is found and accepted according to the SA-based criterion by o_{acc} . The search process of the ALNS is divided into a number of segments of γ iterations. After γ iterations, the new weight of operator $i \in \mathcal{X}$ is calculated as $w_{i+1} = (1-r) \cdot w_i + r \cdot \frac{o_i}{\beta_i}$, where w_i corresponds to the weight of operator $i \in \mathcal{X}$. The speed of the weight adjustment is controlled by reaction parameter $r \in [0, 1]$, which takes the success of an operator in previous segments into account. The number of times that an operator $i \in \mathcal{X}$ was chosen in the previous segment is denoted by β_i . Subsequently, we calculate the probability π_i to choose an operator $i \in \mathcal{X}$ according to $\pi_i = w_i / \sum_{i \in \mathcal{X}} w_i$.

Acceptance criterion We use an acceptance criterion based on SA in order to overcome local optima. An improving solution is always accepted. A new deteriorating solution s' is accepted with probability $e^{-(f(s')-f(s))/t_i}$, where $t_i > 0$ is the current temperature. The starting temperature t_0 is determined such that a solution that deteriorates the current solution by $a\%$ is accepted with a probability of 50%. We decrease the temperature by a constant factor c each time a deteriorating solution is accepted such that in the last 20% of iterations the temperature is below 0.0001.

Finally, we add a noise term η to the objective function according to Ropke and Pisinger (2006a) in order to further diversify the search.

3.2.3 Tabu search component

TS is a local search-based metaheuristic that was originally proposed by Glover (1986). At each iteration, TS examines the neighborhood $\mathcal{N}(s)$ of a current solution s . The solution s is modified by a set of neighborhood operators, called moves. Our TS uses the neighborhood structures $\mathcal{N}_{\text{shift}}$ and $\mathcal{N}_{\text{swap}}$ as described in Section 2.5.4.2 and selects the best neighbor of the current solution in the composite neighborhood $\mathcal{N}(s) = \mathcal{N}_{\text{swap}}(s) \cup \mathcal{N}_{\text{shift}}(s)$, even if the best neighbor is of lower quality than the current solution. Recall that the shift operator generates a neighborhood by shifting a customer order from one batch to another batch and the swap operator by swapping two customer orders between two batches. To avoid short term cycling of the search process, recently visited solutions are prohibited and inserted in a tabu list for ϑ iterations, called tabu tenure. ϑ is randomly drawn from the interval $[\vartheta_{\min}, \vartheta_{\max}]$. Finding a new best overall solution is used as aspiration criterion.

3.3 Computational studies

This section is devoted to assess the design and performance of ALNS \times TS. Section 3.3.1 details the parameter setting of ALNS \times TS. In Section 3.3.2, we introduce the benchmark sets adopted from Henn and Wäscher (2012) and the newly generated large-scale instances. Section 3.3.3 investigates the scaling behavior of our algorithm. The influence of combining ALNS and TS on both solution quality and runtime is studied in Section 3.3.4. Finally, Section 3.3.5 evaluates the performance of ALNS \times TS in comparison to the best-performing OBP methods from the literature.

3.3.1 Parameter setting

For tuning the parameters of ALNS \times TS, we adopt the procedure described in Ropke and Pisinger (2006a). Starting with a good parameter setting that we obtained during the testing phase of our algorithm, we refine the value of a single parameter while keeping the rest of the parameters fixed and perform five runs on the benchmark instances. We choose the parameter setting with the best average result as the final setting for the respective parameter, and we repeat this procedure with the next parameter. Note that we randomly determine the order in which the tuning of the parameters is performed. We found the following parameter setting, which is used

for all computational studies.

In preliminary studies, we observed that the increase of scores in the adaptive weight adjustment should be merit-based. Finding a new overall best solution should receive a stronger reward than finding an arbitrary improving solution. Moreover, we found that the scores should be close to each other. Finally, our preliminary studies show that it is advantageous if the initial weights of the operators $i \in \mathcal{X}$ are low in comparison to the scores o_i . For these reasons, we decided for the following parameter setting: We set the scores used in the adaptive weight adjustment to $o_{best} = 120$, $o_{imp} = 100$, and $o_{acc} = 80$. The initial weight of all heuristics amounts to $\omega = 10$. Furthermore, we set the number of iterations after which the weights of ALNS are adjusted to $\gamma = 100$, the reaction factor to $\epsilon = 0.4$, the noise term to $\eta = 0.05$, the randomization parameter to $p = 3$, and the percentage of deterioration to $a = 25\%$. For TS, we set the tabu tenure to a random value in the interval $[\vartheta_{min} = \frac{1}{5} \cdot n, \vartheta_{max} = 2 \cdot n]$ for each move inserted into the tabu list. In preliminary studies, we found that the number of customer orders removed from the current solution q^- strongly influences the performance of the ALNS and should depend on the number of customer orders n . Therefore, we randomly choose q^- from the interval $[q_{min}^- = 0.175 \cdot n, q_{max}^- = 0.35 \cdot n]$.

In order to achieve a good trade-off between the solution quality and the runtime of our algorithm, we set the number of ALNS iterations to $10 \cdot n$. Higher numbers of iterations slightly improve the average solution quality but lead to significantly higher runtimes. TS is run for $30 \cdot n$ iterations after $0.3 \cdot n$ iterations of ALNS. Table 3.2 summarizes the parameter setting of ALNS \times TS.

All experiments are conducted on a desktop computer with an Intel Core i7-3770 Processor at 3.5 GHz, 16 GB of memory, running Windows 7 Professional. ALNS \times TS is implemented in Java.

3.3.2 Benchmark instances

The performance of our method is assessed on the instances proposed by Henn and Wäscher (2012), which are described in the following. The benchmark instances assume a single-block parallel-aisle warehouse, in which 900 different items are stored in ten parallel picking aisles with 90 storage locations, 45 on the left and 45 on the right of each picking aisle. The depot is located below the entry of the leftmost picking aisle in the front cross aisle.

Parameter	Parameter value
ALNS component:	
γ	100
q_{min}^-, q_{max}^-	$0.175 \cdot n, 0.35 \cdot n$
ϵ	0.4
η	0.05
$o_{best}, o_{imp}, o_{acc}$	120, 100, 80
ω	10
p	3
a	25%
TS component:	
$\vartheta_{min}, \vartheta_{max}$	$\frac{1}{5} \cdot n, 2 \cdot n$

Table 3.2: Overview of the parameter setting of ALNS×TS used in the computational studies.

The physical dimensions of the warehouse are defined as follows: The distance between the depot and the first storage location in the first (i.e., leftmost) picking aisle amounts to 1.5 length units (LUs). Order picking is assumed in the middle of each storage location. A storage location has a length of 1 LU. When leaving a picking aisle, the order picker moves 1 LU in vertical direction. The distance between two picking aisles amounts to 5 LUs. The total length an order picker needs to travel in order to entirely traverse a picking aisle is 47 LUs. The tour of an order picker through the warehouse is determined according to the S-shape or the largest gap picker routing strategy.

The benchmark instances assume two different demand scenarios, uniformly distributed demand (UDD) and class-based demand (CBD). For CBD, three classes with high (A), medium (B), and low (C) demand frequencies are defined. In class A, 10% of the items account for 52% of the demand, in class B, 30% of the items account for 36% of the demand, and in class C, 60% of the items account for 12% of the demand. Items are assigned to storage locations according to demand frequencies. Items of class A are stored in the leftmost picking aisle, items of class B in picking aisles # 2 to # 4, and items of class C in picking aisles # 5 to # 10.

The instances consider different numbers of customer orders $n \in \{40, 60, 80, 100\}$, different capacities (defined in item units) of the picking device $c \in \{30, 45, 60, 75\}$, and a uniformly distributed number of items per customer order, which is randomly

drawn from the interval $[5, 25]$. Four different benchmark sets are defined: UDD/S-shape, UDD/largest gap, CBD/S-shape, and CBD/largest gap. Each benchmark set is grouped into 16 different classes that are identified by the number of customer orders and the capacity of the picking device. Each class contains 40 instances. This leads to $16 \cdot 40 = 640$ instances per benchmark set and thus 2560 benchmark instances in total.

In addition, we generate a set of large-scale OBP instances. These instances are based on the structure of the benchmark instances proposed by Henn and Wäscher (2012). The set contains instances with $n \in \{200, 300, 400, 500, 600\}$ customer orders, different capacities of the picking device $c \in \{6, 9, 12, 15\}$, and a uniformly distributed number of items per customer order, which is randomly drawn from the interval $[1, 5]$. The following instance classes (n, c) defined by the number of customer orders n and the capacity of the picking device c are considered: $(200, 6)$, $(200, 9)$, $(200, 12)$, $(200, 15)$, $(300, 6)$, $(400, 6)$, $(500, 6)$, $(600, 6)$. Each instance class contains 10 instances. We make all instances available for download at <https://www.dropbox.com/sh/89ishzp9o4a9jcf/AABM9m5qOHUUC35OatPMrpMFa?dl=0>.

3.3.3 Scaling behavior of the metaheuristic hybrid

In order to evaluate the performance of our algorithm on large-scale instances, we investigate different parameter configurations of ALNS \times TS to achieve a good trade-off between runtime and solution quality. During preliminary tests, we observed that the number of TS iterations has the strongest impact on solution quality and runtimes. Therefore, we study the following three configurations. The parameters of ALNS \times TS 1, ALNS \times TS 2, and ALNS \times TS 3 are all equal to those of ALNS \times TS except for the number of TS iterations. Here, we set the number of iterations as follows: In ALNS \times TS 1, TS is run for $20 \cdot n$ iterations, in ALNS \times TS 2, TS is run for $25 \cdot n$ iterations, and in ALNS \times TS 3, TS is run for $30 \cdot n$ iterations.

Table 3.3 presents an aggregate view on the performance of the different ALNS \times TS configurations on the large-scale instances described above and reports averages for groups of instances defined by the number of customer orders (column # orders) and the carrying capacity of the picking device (column C). All reported results are based on five runs. In column *BKS*, we provide the best known solution (BKS) as the average of the best objective function values obtained by one of the tested ALNS \times TS configurations for each of the individual instances in the instance group

over the five runs. For each of the ALNS×TS configurations, we give the following information:

- Δ_f^b (%) denotes the average of the percentage gaps of the best objective function value achieved with the respective ALNS×TS configuration to the best objective function value of any of the ALNS×TS configurations over the five runs. For each instance group, the smallest gap found by any of the ALNS×TS configurations is indicated in bold.
- Δ_f^a (%) denotes the average of the percentage gaps of the average objective function value achieved with the respective ALNS×TS configuration to the best objective function value of any of the ALNS×TS configurations over the five runs.
- t (s) reports the average runtime in seconds over the five runs.

We observe that the solution quality improves from ALNS×TS 1 to ALNS×TS 2 to ALNS×TS 3. On average, ALNS×TS 3 deviates by 0.1% from the best solutions obtained with the tested configurations. Furthermore, ALNS×TS 3 is very robust, which can be seen from the small difference between average and best solution quality. With respect to scaling behavior, we find that for $n \geq 300$ adding 100 customer orders approximately doubles the average computation time. Although ALNS×TS 3 uses more time than the other two configurations, we believe that the average runtimes of about 1.5 hours for 600 customer orders are reasonable. In our opinion, ALNS×TS 3 shows the best trade-off between solution quality and runtime and will be used for all further studies.

3.3.4 Effect of algorithmic components

In Table 3.4, we present the performance of ALNS and TS as standalone methods in comparison to ALNS×TS on the benchmark set UDD/S-shape and report averages for groups of instances defined by the number of customer orders (column # orders) and the carrying capacity of the picking device (column C). In column BKS , we provide the BKS as the average of the best objective function values obtained by one of the methods for each of the individual instances in the group. The BKS is indicated by an asterisk if it refers to the optimal solution obtained with commercial solver Gurobi. Otherwise, the BKS refers to the best solution obtained by ALNS×TS. For all methods, we give the following information:

# orders	C	BKS	ALNS×TS 1			ALNS×TS 2			ALNS×TS 3		
			Δ_f^b (%)	Δ_f^a (%)	t (s)	Δ_f^b (%)	Δ_f^a (%)	t (s)	Δ_f^b (%)	Δ_f^a (%)	t (s)
200	6	18376	0.3	0.5	153	0.0	0.3	194	0.1	0.3	224
200	9	12851	0.7	1.4	172	0.4	0.9	222	0.1	0.6	254
200	12	10622	1.2	1.8	197	0.3	1.1	259	0.2	0.7	299
200	15	9007	1.4	2.0	222	0.6	1.4	282	0.1	0.5	325
300	6	28411	0.2	0.4	520	0.1	0.3	653	0.1	0.3	764
400	6	37400	0.1	0.4	1187	0.1	0.3	1522	0.1	0.3	1763
500	6	46604	0.2	0.3	2335	0.0	0.2	2982	0.1	0.3	3423
600	6	55789	0.1	0.3	4166	0.1	0.3	4910	0.1	0.3	5661
Average			0.5	0.9	1119	0.2	0.6	1378	0.1	0.4	1589
Minimum			0.1	0.3	153	0.0	0.2	194	0.1	0.3	224
Maximum			1.4	2.0	4166	0.6	1.4	4910	0.2	0.7	5661

Table 3.3: Results of different ALNS×TS configurations. In the first two columns, we specify the groups of instances defined by the number of customer orders (column # orders) and the carrying capacity of the picking device (column C). In column BKS , we provide the BKS as the average of the best objective function values obtained by one of the tested ALNS×TS configurations for each of the individual instances in the instance group over the five runs. For each of the ALNS×TS configurations, the table reports the average of the percentage gaps of the best objective function value achieved with the respective ALNS×TS configuration to the best objective function value of any of the ALNS×TS configurations (column Δ_f^b (%)), the same measure based on the average solution quality of the respective configuration (column Δ_f^a (%)), and the average runtime in seconds (column t (s)). For each instance group, the smallest gap (Δ_f^b (%)) found by any of the ALNS×TS configurations is indicated in bold.

- Δ_f (%) denotes the percentage gap between the best solution found by the respective method and the BKS. For each group of instances, the smallest gap found by any of the methods is indicated in bold.
- t (s) reports the average runtime in seconds.

It can be observed that pure ALNS and pure TS perform significantly worse in comparison to ALNS×TS. On average, pure ALNS deviates by 2.5%, and pure TS by 1.0% from the solutions of the hybrid. This can be explained by the fact that ALNS has very good diversification capabilities but lacks with respect to intensification, whereas the diversification possibilities of TS seem to be insufficient to address the standard OBP. The results show that achieving good solution quality on the standard OBP requires both intensification and diversification during the search

process.

# orders	C	BKS	ALNS×TS		ALNS		TS	
			Δ_f (%)	t (s)	Δ_f (%)	t (s)	Δ_f (%)	t (s)
40	30	10462*	0.0	2	0.4	1	0.1	4
40	45	6865	0.0	2	1.3	1	0.9	3
40	60	5277	0.0	2	1.7	2	1.2	5
40	75	4274	0.0	3	1.8	2	1.3	5
60	30	15482*	0.1	5	0.8	4	0.2	10
60	45	10035	0.0	6	2.7	5	1.1	11
60	60	7710	0.0	7	2.7	5	1.3	13
60	75	6306	0.0	8	2.6	6	1.1	15
80	30	20645*	0.1	12	1.3	10	0.3	24
80	45	13334	0.0	13	3.6	10	1.6	26
80	60	10175	0.0	16	3.7	13	1.1	32
80	75	8245	0.0	17	3.5	14	1.1	34
100	30	25540*	0.2	23	1.8	18	0.3	45
100	45	16353	0.0	23	4.0	19	1.4	46
100	60	12477	0.0	28	4.5	23	1.2	57
100	75	10154	0.0	33	4.3	26	1.3	65
Average			0.0	12	2.5	10	1.0	25
Minimum			0.0	2	0.4	1	0.1	3
Maximum			0.2	33	4.5	26	1.6	65

Table 3.4: Performance of ALNS and TS as standalone methods in comparison to ALNS×TS on the benchmark set UDD/S-shape. In the first two columns, we detail the groups of instances defined by the number of customer orders (column $\#$ orders) and the carrying capacity of the picking device (column C). In column BKS , we provide the BKS as the average of the best objective function values obtained by one of the methods for each of the individual instances in the instance group. The BKS is indicated by an asterisk if it refers to the optimal solution obtained with Gurobi. Otherwise, the BKS refers to the best solution obtained by ALNS×TS. For all methods, the table reports the percentage gap between the best solution found by the respective method and the BKS (column Δ_f (%)) and the average runtime in seconds (column t (s)). For each instance group, the smallest gap found by any of the methods is indicated in bold.

3.3.5 Comparison to the state-of-the-art

In this section, we compare the solution quality and runtime of ALNS×TS to the best-performing OBP methods from the literature on the benchmark sets described in Section 3.3.2. For all instance classes with a picking device capacity of 30, we solved the model presented in Section 3.1 using Gurobi.

Performance on UDD/S-shape On the benchmark set UDD/S-shape, the performance of ALNS×TS is compared to the methods of Henn et al. (2010) and Henn and Wäscher (2012): local search (LS), ILS-1, ILS-2, TS_{AP}, TS_{BI}, ABHC_{o,o} and ABHC_{b,o}. Henn et al. (2010) and Henn and Wäscher (2012) do not give information about the number of test runs conducted to achieve their results. To make the comparison as fair as possible, we conduct only a single ALNS×TS run per instance.

In Table 3.5, we compare the performance of ALNS×TS to all comparison methods on the benchmark set UDD/S-shape and report averages for groups of instances defined by the number of customer orders (column *# orders*) and the carrying capacity of the picking device (column *C*). In column *BKS*, we provide the BKS as the average of the best objective function values obtained by one of the methods for each of the individual instances in the group. The BKS is indicated by an asterisk if it refers to the optimal solution obtained with Gurobi. Otherwise, the BKS refers to the best solution obtained by one of the comparison methods. For ALNS×TS and all comparison methods, we report the percentage gap to the BKS (column Δ_f (%)). For each instance group, the smallest gap found by any of the methods is indicated in bold. Moreover, the table reports the average runtimes in seconds (column *t* (s)) and the corrected average runtimes in seconds in brackets. The corrected average runtimes take into account that different computers were used to conduct the tests. To make the times comparable, we use the Passmark (PM, see www.passmark.com) score of a single core of the used processors (PM score of AMD Athlon 3500+, 2.21 GHz: 667; PM score of our Intel Core i7, 3.5 GHz: 1853). Note that an entirely fair comparison of runtimes is never possible because programming languages and operating systems may differ.

ALNS×TS beats the solution quality of the best comparison method ABHC_{o,o} on all tested instance classes. Compared to ABHC_{o,o}, ALNS×TS reduces the runtime by approximately 55% on average.

Besides solution quality and runtime, scalability and robustness are important criteria for the evaluation of an algorithm. ALNS×TS achieves the strongest speed-up compared to ABHC_{o,o} for the largest instance classes with $n = 80, 100$ and $c = 30, 45, 60, 75$. This indicates a superior scaling behavior of our algorithm, which may be quite significant in practice. Moreover, ALNS×TS shows a more robust performance with a maximum gap to the BKS of 0.14%, whereas the gaps of ABHC_{o,o} fluctuate between 0.14% and 1.03%.

# orders	C	BKS	LS		ILS-1		ILS-2		TSAP		TSBI		ABHC _{o,o}		ABHC _{b,o}		ALNS \times TS	
			Δ_f (%)	t (s)	Δ_f (%)	t (s)	Δ_f (%)	t (s)	Δ_f (%)	t (s)	Δ_f (%)	t (s)	Δ_f (%)	t (s)	Δ_f (%)	t (s)	Δ_f (%)	t (s)
40	30	10462*	2.76	0 (0)	0.90	3 (1)	0.76	3 (1)	1.38	2 (1)	1.33	4 (1)	0.28	2 (1)	0.21	2 (1)	0.00	2
40	45	6858	5.60	0 (0)	1.38	7 (2)	1.61	7 (2)	0.75	4 (1)	1.43	7 (2)	0.44	4 (1)	1.07	2 (1)	0.03	2
40	60	5271	4.74	0 (0)	1.27	7 (2)	1.34	8 (3)	0.75	5 (2)	1.13	8 (3)	0.64	4 (1)	1.52	2 (1)	0.07	2
40	75	4268	4.80	0 (0)	1.99	6 (2)	1.73	8 (3)	0.87	6 (2)	1.55	10 (4)	1.03	4 (1)	1.83	1 (0)	0.02	3
60	30	15482*	2.93	0 (0)	0.69	15 (5)	0.77	16 (6)	1.26	6 (2)	2.09	13 (5)	0.19	10 (4)	0.29	13 (5)	0.03	5
60	45	10018	4.60	0 (0)	1.59	34 (12)	1.46	34 (12)	0.63	11 (4)	1.33	22 (8)	0.43	23 (8)	0.69	13 (5)	0.07	6
60	60	7692	5.10	0 (0)	1.97	31 (11)	1.89	38 (14)	0.90	13 (5)	1.52	27 (10)	0.73	25 (9)	1.39	11 (4)	0.09	7
60	75	6286	4.37	0 (0)	1.51	28 (10)	1.72	40 (14)	0.90	15 (5)	1.21	38 (14)	0.81	29 (10)	1.25	9 (3)	0.05	9
80	30	20645*	3.03	0 (0)	0.71	50 (18)	0.86	49 (18)	1.46	12 (4)	1.99	32 (11)	0.14	37 (13)	0.25	54 (19)	0.04	12
80	45	13310	4.08	0 (0)	1.62	100 (36)	1.66	103 (37)	0.68	21 (8)	1.41	50 (18)	0.28	82 (30)	0.62	44 (16)	0.08	13
80	60	10159	4.36	0 (0)	1.62	95 (34)	1.49	111 (40)	0.69	24 (9)	1.28	70 (25)	0.27	97 (35)	0.78	40 (14)	0.09	16
80	75	8223	4.63	0 (0)	1.51	100 (36)	1.93	108 (39)	0.83	29 (10)	1.08	79 (29)	0.57	106 (38)	1.12	33 (12)	0.04	20
100	30	25540*	3.34	0 (0)	0.81	126 (45)	0.90	121 (43)	1.29	20 (7)	2.54	57 (21)	0.16	111 (40)	0.27	162 (58)	0.08	21
100	45	16321	4.06	1 (0)	1.99	222 (80)	1.96	250 (90)	0.78	39 (14)	1.68	98 (35)	0.23	207 (75)	0.52	111 (40)	0.14	23
100	60	12455	4.47	1 (0)	1.66	222 (80)	2.11	237 (85)	0.67	43 (15)	1.10	127 (46)	0.21	253 (91)	0.80	96 (35)	0.09	30
100	75	10126	4.81	1 (0)	2.25	238 (86)	2.47	276 (99)	0.91	55 (20)	1.58	156 (56)	0.36	280 (101)	0.99	84 (30)	0.13	37
Average			4.23	0 (0)	1.47	80 (29)	1.54	88 (32)	0.92	19 (7)	1.52	50 (18)	0.42	80 (29)	0.85	42 (15)	0.07	13
Minimum			2.76	0 (0)	0.69	3 (1)	0.76	3 (1)	0.63	2 (1)	1.08	4 (1)	0.14	2 (1)	0.21	1 (0)	0.00	2
Maximum			5.60	1 (0)	2.25	238 (86)	2.47	276 (99)	1.46	55 (20)	2.54	156 (56)	1.03	280 (101)	1.83	162 (58)	0.14	37

Table 3.5: Performance of ALNS \times TS on the benchmark set UDD/S-shape. In the first two columns, we detail the groups of instances defined by the number of customer orders (column # orders) and the carrying capacity of the picking device (column C). In column BKS , we provide the BKS as the average of the best objective function values obtained by one of the methods for each of the individual instances in the group. The BKS is indicated by an asterisk if it refers to the optimal solution obtained with Gurobi. Otherwise, the BKS refers to the best solution obtained by one of the comparison methods. For ALNS \times TS and all comparison methods, we report the percentage gap between the best solution found by the respective method and the BKS (column Δ_f (%)), the average runtime in seconds (column t (s)), and the corrected average runtime in seconds in brackets. For each instance group, the smallest gap found by any of the methods is indicated in bold.

Performance on CBD/S-shape We contacted the authors of Henn and Wäscher (2012) regarding the objective function values of their methods on the individual instance classes of the benchmark set CBD/S-shape, which were used in the papers of Henn et al. (2010) and Henn and Wäscher (2012). Unfortunately, these results are no longer available from the authors. As an alternative, we compare ALNS×TS to IGA and GGA proposed by Koch and Wäscher (2016). Here, again no information is given about the number of test runs conducted to achieve the results, and we use the results of a single ALNS×TS run for comparison.

Table 3.6 reports the performance of ALNS×TS in comparison to the different GA variants on the benchmark set CBD/S-shape. With respect to solution quality, ALNS×TS provides near-optimal solutions for picking device capacities of 30 items; the average gap to the exact solutions stays below 0.1%. Moreover, ALNS×TS outperforms all GA variants on all instance classes. On average, ALNS×TS improves the solution quality of GGA by 1.7% and that of IGA by 2.4%, while reducing the runtime by nearly 90%. On the largest instances reported ($n = 60$ and $c = 75$), the improvement of solution quality obtained by ALNS×TS is strongest (gap of 5.8% to IGA and of 5.1% to GGA). This again indicates a better suitability of ALNS×TS to address larger problem instances. It is quite likely that the superiority of our method would increase with instance size, however, Koch and Wäscher (2016) do not report results for the larger instances with $n = 80, 100$.

Performance on UDD/largest gap and CBD/largest gap We found that our computation of the distance according to the largest gap picker routing strategy differs from the computation of Henn et al. (2010) and Henn and Wäscher (2012). For instances with a picking device capacity of 30 items, we find optimal solutions that are superior to the optimal solutions found by Henn et al. (2010) and Henn and Wäscher (2012), which proves a different interpretation of the largest gap picker routing strategy.

Therefore, in Table 3.7, the performance of ALNS×TS is compared to the results of C&W(ii) on the benchmark sets UDD/largest gap and CBD/largest gap. The BKS refers to the optimal solution obtained with Gurobi for the instances with a picking device capacity of 30 items, for all other instances to the best result obtained with ALNS×TS.

Again, it can be observed that ALNS×TS provides near-optimal solutions for the classes with small capacity of the picking device. Compared to C&W(ii),

# orders	C	BKS	IGA		GGA		ALNS×TS	
			Δ_f (%)	t (s)	Δ_f (%)	t (s)	Δ_f (%)	t (s)
20	30	4192*	0.3	3 (1)	0.1	7 (3)	0.0	0
20	45	2688	2.1	4 (2)	0.4	8 (3)	0.0	0
20	60	2155	1.4	5 (2)	0.6	9 (3)	0.0	0
20	75	1742	1.5	5 (2)	0.8	10 (4)	0.0	0
30	30	6188*	0.6	9 (3)	0.1	17 (6)	0.0	1
30	45	4080	1.8	15 (5)	1.0	21 (7)	0.0	1
30	60	3079	2.3	17 (6)	1.5	25 (9)	0.0	1
30	75	2530	2.9	17 (6)	2.1	27 (10)	0.0	1
40	30	7907*	0.5	23 (8)	0.2	35 (12)	0.0	2
40	45	5183	2.4	40 (14)	1.7	45 (16)	0.0	2
40	60	3986	3.2	49 (18)	2.6	54 (19)	0.0	2
40	75	3243	4.1	50 (18)	3.2	60 (22)	0.0	3
50	30	10098*	0.9	47 (17)	0.3	61 (22)	0.0	4
50	45	6462	2.3	87 (31)	1.8	86 (31)	0.0	4
50	60	4988	3.3	117 (42)	2.7	109 (39)	0.0	5
50	75	4077	4.3	109 (39)	3.5	117 (42)	0.0	6
60	30	11609*	1.0	95 (34)	0.6	100 (36)	0.1	5
60	45	7550	3.3	175 (63)	2.7	150 (54)	0.0	5
60	60	5819	4.6	230 (83)	3.7	187 (67)	0.0	7
60	75	4724	5.8	242 (87)	5.1	208 (75)	0.0	8
Average			2.4	67 (24)	1.7	67 (24)	0.0	3
Minimum			0.3	3 (1)	0.1	7 (3)	0.0	0
Maximum			5.8	242 (87)	5.1	208 (75)	0.1	8

Table 3.6: Performance of ALNS×TS on the benchmark set CBD/S-shape. In the first two columns, we detail the groups of instances defined by the number of customer orders (column *# orders*) and the carrying capacity of the picking device (column *C*). In column *BKS*, we provide the BKS as the average of the best objective function values obtained by one of the methods for each of the individual instances in the group. The BKS is indicated by an asterisk if it refers to the optimal solution obtained with Gurobi. Otherwise, the BKS refers to the best solution obtained by one of the comparison methods. For ALNS×TS and all comparison methods, we report the percentage gap between the best solution found by the respective method and the BKS (column Δ_f (%)), the average runtime in seconds (column t (s)), and the corrected average runtime in seconds in brackets. For each instance group, the smallest gap found by any of the methods is indicated in bold.

# orders	C	UDD/largest gap				CBD/largest gap			
		BKS	$\Delta_f^{C\&W(ii)}$ (%)	$\Delta_f^{ALNS \times TS}$ (%)	t (s)	BKS	$\Delta_f^{C\&W(ii)}$ (%)	$\Delta_f^{ALNS \times TS}$ (%)	t (s)
40	30	9679*	2.7	0.0	5	6940*	3.8	0.1	5
40	45	7111	4.7	0.0	8	5048	4.4	0.0	8
40	60	5603	6.3	0.0	12	3945	5.5	0.0	13
40	75	4815	7.4	0.0	18	3397	6.6	0.0	18
60	30	14873*	2.6	0.0	17	10671*	4.2	0.1	17
60	45	10400	4.2	0.0	26	7377	4.0	0.0	28
60	60	8184	6.3	0.0	39	5749	5.1	0.0	42
60	75	7101	7.5	0.0	47	4965	6.6	0.0	51
80	30	19223*	3.0	0.3	36	13737*	4.6	0.3	34
80	45	13569	4.2	0.0	57	9559	3.4	0.0	60
80	60	11121	5.7	0.0	86	7753	4.5	0.0	89
80	75	9289	7.2	0.0	109	6464	5.9	0.0	118
100	30	23898*	2.9	0.3	73	17095*	4.7	0.3	67
100	45	16934	3.7	0.0	112	11831	3.4	0.0	115
100	60	13541	5.6	0.0	152	9423	4.4	0.0	165
100	75	12034	7.0	0.0	197	7837	5.8	0.0	221
Average			5.1	0.0	62		4.8	0.0	66
Minimum			2.6	0.0	5		3.4	0.0	5
Maximum			7.5	0.3	197		6.6	0.3	221

Table 3.7: Performance of ALNS \times TS on the benchmark sets UDD/largest gap and CBD/largest gap. In the first two columns, we detail the groups of instances defined by the number of customer orders (column # orders) and the carrying capacity of the picking device (column C). In column BKS , we provide the BKS as the average of the best objective function values obtained by one of the methods for each of the individual instances in the group. The BKS is indicated by an asterisk if it refers to the optimal solution obtained with Gurobi. Otherwise, the BKS refers to the best solution obtained by ALNS \times TS. For C&W(ii) and ALNS \times TS, we provide the percentage gap between the best solution found by the respective method and the BKS (columns $\Delta_f^{C\&W(ii)}$ (%) and $\Delta_f^{ALNS \times TS}$ (%)) and the average runtime in seconds (column t (s)). For each instance group, the smallest gap found by any of the methods is indicated in bold.

ALNS×TS improves the total tour length by 5.1% (UDD/largest gap) and by 4.8% (CBD/largest gap) on average. On both benchmark sets, ALNS×TS provides stronger improvements for large capacities of the picking device. With respect to runtime, the table shows that the largest gap routing requires more runtime than the S-shape routing. This can be explained by the fact that for each picking aisle the gaps between all picking locations have to be computed.

Performance in comparison to Öncan (2015) In the following, we compare our ALNS×TS to the ILST proposed by Öncan (2015). Note that Öncan (2015) calculate the length for entering a picking aisle to 0.5 LUs, and the distance between the depot and the front cross aisle is included. Contrary to this, in Henn et al. (2010), Henn and Wäscher (2012), Koch and Wäscher (2016), and in our dissertation, the length is set to 1 LU, and the distance between the depot and the front cross aisle is neglected. To make results comparable, we have adapted our calculation to match the one of Öncan (2015).

Table 3.8 reports the performance of ALNS×TS in comparison to ILST on the benchmark set UDD/S-shape. Here, ALNS×TS beats the solution quality of ILST on 17 out of 20 instance classes and matches it on two instance classes. Only on one instance class, ALNS×TS is slightly outperformed by ILST. The average improvement of ALNS×TS is 1.2%. At the same time, ALNS×TS is able to reduce the runtime by approximately 80%. We note that for the instances with a larger number of customer orders ($n = 80, 100$) and larger capacities of the picking device ($c = 45, 60, 75$), ALNS×TS provides clearly superior solution quality (with improvements of up to 4.3% for the individual instance classes). This indicates a better suitability of ALNS×TS to address larger problem instances.

On the benchmark set CBD/S-shape (see Table 3.9), ALNS×TS beats ILST on 10 out of 20 instance classes and matches it on 8 instance classes. While ILST deviates by 1.3% from the BKS on average, ALNS×TS shows only a small gap of 0.1% to the BKS and again, the results of ALNS×TS for the larger instances are significantly superior (improvements of up to 5.3% for the individual instance classes). ALNS×TS is again able to reduce runtimes by approximately 80%.

# orders	C	BKS	ILST		ALNS \times TS	
			Δ_f (%)	t (s)	Δ_f (%)	t (s)
20	30	5565*	0.0	2 (1)	0.0	0
20	45	3487	0.2	1 (1)	0.0	0
20	60	2739	0.2	2 (1)	0.0	0
20	75	2227	0.1	1 (1)	0.0	1
40	30	10294*	0.1	9 (6)	0.0	2
40	45	6744	0.9	10 (7)	0.0	2
40	60	5187	0.3	12 (8)	0.0	2
40	75	4200	0.1	10 (7)	0.0	2
60	30	15234*	0.0	17 (12)	0.0	5
60	45	9877	0.8	21 (15)	0.0	6
60	60	7584	1.2	18 (13)	0.0	7
60	75	6196	1.0	19 (14)	0.0	8
80	30	20316*	0.0	75 (54)	0.1	12
80	45	13139	2.1	74 (54)	0.0	13
80	60	10013	2.9	63 (46)	0.0	16
80	75	8114	3.2	51 (37)	0.0	17
100	30	25132*	0.2	332 (240)	0.1	22
100	45	16101	2.5	316 (229)	0.0	24
100	60	12277	3.6	323 (234)	0.0	28
100	75	10002	4.3	279 (202)	0.0	33
Average			1.2	82 (59)	0.0	10
Minimum			0.0	1 (1)	0.0	0
Maximum			4.3	332 (240)	0.1	33

Table 3.8: Performance of ALNS \times TS in comparison to ILST on the benchmark set UDD/S-shape. In the first two columns, we detail the groups of instances defined by the number of customer orders (column # orders) and the carrying capacity of the picking device (column C). In column BKS , we provide the BKS as the average of the best objective function values obtained by one of the methods for each of the individual instances in the group. The BKS is indicated by an asterisk if it refers to the optimal solution obtained with Gurobi. Otherwise, the BKS refers to the best solution obtained by one of the comparison methods. For ALNS \times TS and ILST, we report the percentage gap between the best solution found by the respective method and the BKS (column Δ_f (%)), the average runtime in seconds (column t (s)), and the corrected average runtime in seconds in brackets. For each instance group, the smallest gap found by any of the methods is indicated in bold.

# orders	C	BKS	ILST		ALNS×TS	
			Δ_f (%)	t (s)	Δ_f (%)	t (s)
20	30	4134*	0.0	1 (1)	0.0	0
20	45	2650	0.0	1 (1)	0.0	0
20	60	2123	0.0	1 (1)	0.0	0
20	75	1717	0.0	1 (1)	0.0	0
40	30	7797*	0.0	9 (7)	0.0	2
40	45	5101	0.0	9 (6)	0.0	2
40	60	3908	0.0	8 (6)	0.4	2
40	75	3173	0.0	8 (6)	0.5	3
60	30	11448*	0.0	19 (13)	0.0	5
60	45	7465	0.6	17 (12)	0.0	6
60	60	5747	0.7	21 (15)	0.0	7
60	75	4667	1.2	19 (14)	0.0	8
80	30	15395*	0.1	52 (38)	0.1	12
80	45	9912	2.1	67 (49)	0.0	13
80	60	7544	3.6	58 (42)	0.0	14
80	75	6130	4.5	70 (51)	0.0	18
100	30	18851*	0.3	287 (208)	0.1	21
100	45	12106	2.8	279 (202)	0.0	23
100	60	9285	4.2	270 (195)	0.0	27
100	75	7557	5.3	295 (214)	0.0	28
Average			1.3	75 (54)	0.1	10
Minimum			0.0	1 (1)	0.0	0
Maximum			5.3	295 (214)	0.5	28

Table 3.9: Performance of ALNS×TS in comparison to ILST on the benchmark set CBD/S-shape. In the first two columns, we detail the groups of instances defined by the number of customer orders (column *# orders*) and the carrying capacity of the picking device (column C). In column *BKS*, we provide the BKS as the average of the best objective function values obtained by one of the methods for each of the individual instances in the group. The BKS is indicated by an asterisk if it refers to the optimal solution obtained with Gurobi. Otherwise, the BKS refers to the best solution obtained by one of the comparison methods. For ALNS×TS and ILST, we report the percentage gap between the best solution found by the respective method and the BKS (column Δ_f (%)), the average runtime in seconds (column t (s)), and the corrected average runtime in seconds in brackets. For each instance group, the smallest gap found by any of the methods is indicated in bold.

3.4 Summary and conclusion

This chapter addresses the standard OBP to optimize the grouping of customer orders to picking orders (batches) with the objective of reducing the total length of order picking tours. To solve larger instances within short runtime, we propose a metaheuristic hybrid based on ALNS and TS.

In numerical studies, we conduct an extensive comparison of ALNS \times TS to all previously published methods that test the performance of their method on the standard OBP benchmark sets from the literature. ALNS \times TS is able to beat any previously proposed method for the standard OBP concerning the average solution quality and runtime over all benchmark sets. For settings with a larger number of customer orders and larger capacities of the picking device, ALNS \times TS shows the clearest advantages compared to the state-of-the-art methods with respect to solution quality.

Furthermore, our approach is able to solve newly generated large instances with up to 600 customer orders and six items per customer order with reasonable runtimes and with good scaling behavior and robustness.

Chapter 4

Picker routing and storage assignment strategies for precedence-constrained order picking

The contents of this chapter are included in similar form in the following publication: I. Žulj, C. H. Glock, E. H. Grosse, and M. Schneider. Picker routing and storage assignment strategies for precedence-constrained order picking. *Computers & Industrial Engineering*, 123:338–347, 2018.

As described before, precedence constraints define that certain items need to be collected before other items. Although many real-world warehouses face such constraints in order picking, they are hardly considered in the warehousing literature. This chapter is devoted to address a precedence-constrained PRP, which is inspired by a practical case observed in a warehouse of a German manufacturer of household products. In this warehouse, order pickers are not permitted to put heavy items on top of light items during picking to prevent damage to the light items.

The remainder of this chapter is organized as follows. In Section 4.1, we give a detailed problem description. To address the problem, we propose a picker routing strategy (E-PRS_W) that incorporates the precedence constraint by picking heavy items before light items in an optimal fashion. Section 4.2 describes our exact algorithm used to evaluate E-PRS_W. In Section 4.3, we present a practical case study that is based on a dataset provided to us by the case company to test E-PRS_W. In the case study, (i) we compare the performance of E-PRS_W to those of other picker routing strategies (H-PRS_{W/O} and E-PRS_{W/O}), (ii) we introduce different

item storage assignment strategies that consider the weight of the items when assigning items to storage locations, and (iii) we examine the impact of these item storage assignment strategies on the three picker routing strategies. Subsequently, Section 4.4 investigates the influence of different problem parameters on the performance of E- PRS_W , namely the warehouse size, the share of heavy and light items per customer order, and the number of requested items per customer order. Moreover, we derive insights for warehouse managers dealing with the given precedence constraint in order picking. The chapter concludes with a summary in Section 4.5.

4.1 Problem description

We consider a single-block parallel-aisle warehouse with a central depot located below the entry of the leftmost picking aisle in the front cross aisle (as described in Section 2.3.1). At the depot, an order picker receives a pick list and a picking device. A pick list specifies a single customer order and contains a non-empty set of order lines, where each order line indicates a particular item and its weight, the requested quantity of this item as well as its storage location in the picking area.

To prevent damage to light items, an order picker is not allowed to put heavy items on top of light items. Thus, heavy items can only be placed above other heavy items, while light items can be placed above heavy items or other light items. We assume that all items of a customer order are collected on a single order picking tour, which starts from the depot, proceeds along the storage locations defined by the respective customer order, and ends at the depot.

To route an order picker through the warehouse while respecting the given precedence constraint, we consider the following three picker routing strategies:

- $H-PRS_{W/O}$: In the case company under study, an order picker's tour through the warehouse (and thus the retrieval sequence of the requested items from their storage locations) is determined by applying a simple S-shape routing strategy that does not consider the precedence constraint. As a result, an order picker collects the items of a customer order in a plastic box without stacking them on top of each other. Upon return to the depot, the collected items are sorted and packed into a cardboard box that is used for shipping the items to the respective customer such that the precedence constraint is respected.
- $E-PRS_{W/O}$: According to E- $PRS_{W/O}$, the items of a customer order are collected

and placed (next to each other) into a plastic box without considering the precedence constraint. Contrary to $H\text{-}PR_{S_{W/O}}$, order picking is carried out in an optimal fashion with respect to the routing of an order picker. At the end of the order picking process, the collected items are sorted and packed in a cardboard box respecting the precedence constraint.

- $E\text{-}PR_{S_W}$: To avoid the sorting of the collected items after the retrieval process, we propose a new picker routing strategy ($E\text{-}PR_{S_W}$) that incorporates the precedence constraint and collects heavy items before light items. Obviously, the collected items can be directly placed in the cardboard box associated with the respective customer. We assume that an order picker follows a one-dimensional stacking procedure when placing the items in the cardboard box.

These picker routing strategies are evaluated with respect to the objective of minimizing the travel distance of an order picker for collecting the items of a customer order and the sorting effort, which arises for $H\text{-}PR_{S_{W/O}}$ and $E\text{-}PR_{S_{W/O}}$.

Of course, it would be possible to consider a hybrid picker routing strategy that combines $E\text{-}PR_{S_{W/O}}$ and $E\text{-}PR_{S_W}$, i.e., a picker routing strategy that determines the optimal retrieval sequence when sorting is carried out while picking. However, such a solution approach is likely to be less useful for practical applications due to the complexity of the resulting order picking process and the high potential for errors: The order picker would have to implement a predefined sorting scheme (due to the one-dimensional stacking system) in addition to traveling on a given route through the warehouse. Furthermore, Elbert et al. (2017) find that order pickers deviate from complex routes (e.g., due to confusion) and thus recommend more straightforward and non-confusing picker routing strategies. In light of these limitations, we refrain from studying such a hybrid picker routing strategy.

4.2 Solution algorithm

In this section, we present an exact algorithm to evaluate $E\text{-}PR_{S_W}$. To this end, the algorithm determines the optimal tour of minimum travel length of an order picker for collecting heavy items before light items of a given customer order on a single order picking tour. Our solution approach is based on the algorithm of Löffler et al. (2020), which is an extension of the algorithm introduced by Ratliff and Rosenthal (1983). Because we do not modify the algorithm of Löffler et al. (2020), we do not

give a description of their algorithm and instead refer the reader to the original work.

Our algorithm can be described as follows: We define two types of subtours, namely heavy subtours and light subtours. A heavy subtour t_i^{heavy} defines an optimal route through the warehouse for collecting all heavy items $i = 1, \dots, H$ of a customer order, starting from the depot and ending at a predetermined heavy item storage location i . A light subtour t_i^{light} defines an optimal route through the warehouse for collecting all light items $j = 1, \dots, L$ of a customer order, starting from the end location i of heavy subtour t_i^{heavy} and ending at the depot.

Note that each location that contains a heavy item to be picked may be the end location of a heavy subtour and the start location of a light subtour that leads to the minimum tour length for collecting heavy and light items in sequence. To illustrate this, Figure 4.1 depicts the resulting order picking tours for different end locations of a heavy subtour and start locations of a light subtour, respectively. We assume that a customer order consists of two heavy items (h_1 and h_2) and a light item (l_1). There are two possible end (start) locations for the heavy (light) subtour. In Figure 4.1(a), heavy item location h_1 is the end location of heavy subtour $t_{h_1}^{heavy}$ and the start location of light subtour $t_{h_1}^{light}$. In Figure 4.1(b), heavy item location h_2 is the end location of heavy subtour $t_{h_2}^{heavy}$ and the start location of light subtour $t_{h_2}^{light}$. The minimum total tour length is realized in Figure 4.1(a) by retrieval sequence h_2, h_1, l_1 .

Because the algorithm of Löffler et al. (2020) allows arbitrary start and end locations of an order picking tour, we use their algorithm to determine the heavy and light subtours. Then, the optimal sequence for retrieving the required items $G = H + L$ from their storage locations is determined by finding a combination of a heavy subtour t_i^{heavy} and a light subtour t_i^{light} that leads to a minimum total tour length $f(t^*)$. Note that the optimality of the picking sequence is guaranteed by evaluating all possible combinations of heavy and light subtours as follows:

$$f(t^*) = \min_i \left\{ f(t_i^{heavy}) + f(t_i^{light}) \right\} \quad (4.1)$$

$f(t_i^{heavy})$ denotes the travel distance for collecting all heavy items of a customer order on heavy subtour t_i^{heavy} , and $f(t_i^{light})$ for collecting all light items of a customer order on light subtour t_i^{light} .

For a single-block parallel-aisle warehouse, our precedence-constrained PRP can

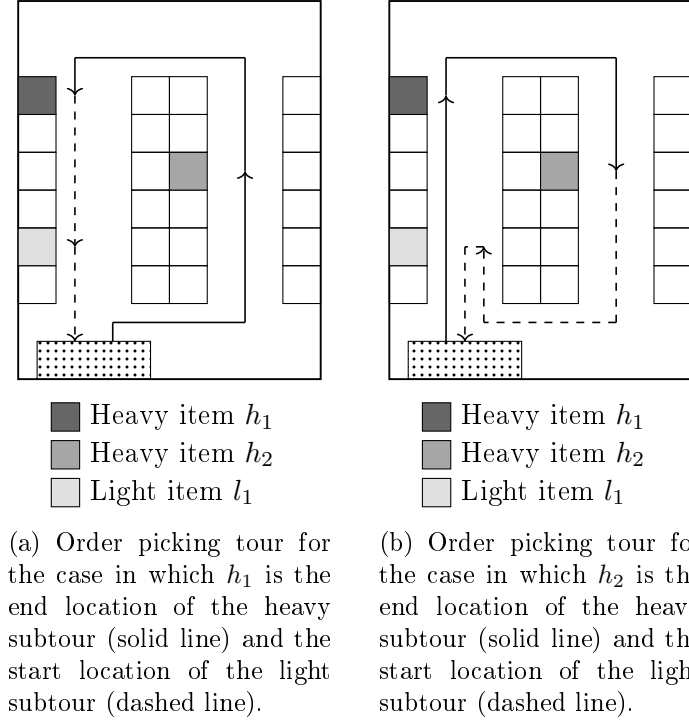


Figure 4.1: Order picking tours for different end locations of a heavy subtour and start locations of a light subtour, respectively.

be solved in polynomial time. The algorithm calls the method of Löffler et al. (2020) for all $H \cdot L$ possible combinations of linking a heavy subtour with a light subtour. Thus, our algorithm has the following runtime complexity:

$$\mathcal{O}((G^3 + P \cdot G^2) \cdot H \cdot L) \approx \mathcal{O}((G^3 + P \cdot G^2) \cdot G^2) \approx \mathcal{O}(G^5),$$

where P denotes the number of picking aisles.

4.3 Practical case study

This section is devoted to assess the performance of the three picker routing strategies within a practical case study. Section 4.3.1 introduces the practical case that motivated the study at hand. Section 4.3.2 evaluates the picker routing strategies, proposes different weight-based storage assignment strategies (W-SASs), and investigates their influence on the performance of the strategies.

4.3.1 Case description

The newly proposed picker routing strategy (E-PRSW) was applied to a scenario motivated by a practical case to investigate the influence of different item weight classes and different storage assignment strategies on the routing of order pickers through a warehouse.

The case company considered here produces household products (e.g., fluid bath additives and natural cosmetics) and operates a distribution warehouse that stores a large variety of items. In the following, we describe the warehouse layout, the physical dimensions of the warehouse, the item weight classes, the generation of pick lists, and the order picking process.

- *Warehouse layout:* For the case study, a simplified model of the real case warehouse was built that consists of a rectangular single-block picking area composed of 10 picking aisles with 100 storage locations per picking aisle (50 storage locations on each side), a cross aisle at the front end of each picking aisle, a cross aisle at the rear end of each picking aisle, and a central depot located below the entry of the leftmost picking aisle in the front cross aisle. The case company does not use a specific dedicated storage assignment strategy but instead assigns items randomly to the storage locations in the warehouse. Each item is available from exactly one storage location.
- *Physical dimensions of the warehouse:* The distance between the depot and the first storage location of the first (i.e., leftmost) picking aisle amounts to 1 LU. A storage location has a length of 1 LU. Picking aisles are narrow enough such that an order picker positioned in the middle of a picking aisle can retrieve items from both sides of the picking aisle without performing additional movements. The distance between two picking aisles is 5 LUs. The total distance an order picker has to cover to entirely traverse a picking aisle is 50 LUs.
- *Item weight classes:* Items stored in the warehouse range from very small glass phials weighing 50 grams up to big wreaths of plastic vessels weighing up to 10 kilograms. Because the items significantly differ in size, weight, and physical features, it is necessary to pack light items on top of heavy items to avoid damage during shipping. An item is categorized as “light” if its weight does not exceed 0.75 kilograms, otherwise as “heavy”. Light and heavy items each account for about 50% of the total number of items in the warehouse.

- *Generation of pick lists:* Based on a dataset provided to us by the case company, a case study instance was generated that consists of 2089 customer orders with 40365 items requested in total, of which 20184 are heavy items and 20181 are light items. A customer requests approximately 19 items on average. Each customer order is given by a single pick list containing a non-empty set of order lines, where each order line specifies a particular item and its weight, the requested quantity of this item as well as its storage location in the warehouse.

Order picking in the warehouse is completely manual, and technical equipment for supporting the order picking process, such as pick-by-light or pick-by-vision, is not available. The order picking process in the case company can be described as follows: At the depot, an order picker receives a paper-based pick list and a standard hand trolley for transporting the requested items through the warehouse. The hand trolley's capacity is sufficient to carry all items contained in a single customer order on a single order picking tour. The S-shape strategy is applied to determine the order picker's tour through the warehouse and thus the sequence for collecting the items of a customer order. The order picker starts from the depot and walks to the storage location of the first item specified by the pick list to retrieve the item in the requested quantity. After having placed the items on the hand trolley (recall that items are placed next to each other), the order picker either proceeds to the next picking location if the pick list contains further items to be collected, or she returns to the depot if there are no further items to be collected. Upon arrival at the depot, the items are sorted and packed in a cardboard box required for shipping to the respective customer such that the precedence constraint is respected. Afterwards, the order picker receives a new pick list and a hand trolley if there are further customer orders to be processed, and the described procedure is repeated.

Note that the sorting and packing of items at the end of the order picking process is very time-consuming in the considered warehouse. During on-site visits, the warehouse manager informed us that the company has tested a sort-while-pick strategy in the past according to which the order pickers already sort items during picking. However, due to the frequent (re-)handling of items, this process proved to be too error-prone in the warehouse at hand.

4.3.2 Results of the case study

The aim of the case study is to compare the current order picking performance in the case company (which induces a high sorting effort) to the performance obtained using the proposed picker routing strategy that integrates the precedence constraint and enables the order picker to pack items directly after retrieving them without additional sorting effort.

For a fair comparison, sorting effort has to be considered by means of penalties when comparing the picker routing strategies. Recall that according to H-PR_{SW/O} and E-PR_{SW/O}, sorting takes place at the end of the order picking process, i.e., all items need to be sorted into cardboard boxes used for shipping the items to the respective customers.

We define different sorting penalty scenarios based on experimental tests that were conducted in the case company. Here, we observed that resorting of items ranges approximately between 3 seconds and 4 seconds per item. This resorting time also includes the time for searching an item in a plastic box and the time for identifying an item as “light” or “heavy” in order to determine the stacking sequence.

Assuming that an order picker’s travel velocity is constant, the travel time is equivalent to the travel distance of all order picking tours (Jarvis and McDowell 1991). Therefore, the resorting time can be added as a sorting penalty measured in LUs to the objective of minimizing the travel distance of an order picker for collecting the items of a customer order. We assume the travel velocity of an order picker to be 1.45 meters per second and define the following scenarios for the sorting effort per item to be resorted: 3 LUs (approximately 1 second), 6 LUs (approximately 2 seconds), 9 LUs (approximately 3 seconds), and 12 LUs (approximately 4 seconds).

Comparison of the picker routing strategies The performance of the picker routing strategies is assessed on the dataset provided to us by the case company as described above. As performance measure for comparing the picker routing strategies, we use the average tour length for collecting the items of all customer orders including the sorting penalty. In Table 4.1, we compare the performance of the picker routing strategies assuming random storage. For all comparison strategies, we report the percentage gap between the best solution found by the respective picker routing strategy and the BKS (column Δ_f (%)) for different sorting efforts (column SE (LUs)). The BKS corresponds to the average of the best objective func-

tion values obtained for each of the single customer orders by one of the tested picker routing strategies. We compute the percentage gap as $\Delta_f = 100 \cdot (f_k - BKS) / BKS$, where f_k denotes the average of the objective function values over the individual instances for picker routing strategy $k \in K$. The smallest average gap found by any of the strategies is indicated in bold. The BKS for each individual instance is available for download at http://www.dpo.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaabajivj. Since the runtime of the proposed algorithm is below one second on all tested instances, we do not explicitly report it.

SE (LUs)	H-PRSW/O	E-PRSW/O	E-PRSW
	Δ_f (%)	Δ_f (%)	Δ_f (%)
0	34.2	0.0	18.8
3	31.1	0.0	8.0
6	29.8	1.0	0.0
9	38.3	9.5	0.0
12	46.7	17.9	0.0

Table 4.1: Performance of the picker routing strategies assuming random storage. For all comparison strategies, we report the percentage gap between the best solution found by the respective picker routing strategy and the BKS (column Δ_f (%)) for different sorting effort scenarios (column SE (LUs)). The BKS corresponds to the average of the best objective function values obtained for each of the single customer orders by one of the tested picker routing strategies. For each sorting effort scenario, the smallest gap found by any of the picker routing strategies is indicated in bold.

The table shows that E-PRSW/O and E-PRSW outperform H-PRSW/O for all sorting effort scenarios with respect to the average total tour length. H-PRSW/O deviates by 29.8% to 46.7% from the BKS for different sorting effort scenarios. Even if no sorting effort is considered (SE = 0), E-PRSW shows a significantly smaller deviation from the BKS compared to H-PRSW/O. If the sorting effort is 6 LUs or higher, E-PRSW outperforms E-PRSW/O.

Effect of different weight-based storage assignment strategies Besides the routing of order pickers, the allocation of items to storage locations in the warehouse influences the resulting tour length of order pickers through the warehouse when collecting the requested items of a customer order.

Obviously, separating heavy items and light items in the warehouse and allocating

heavy items close to the depot is in favour of $E\text{-PRS}_W$ because heavy items are collected before light items. Therefore, different W-SASs are proposed in the following, and their performance in combination with the presented picker routing strategies is evaluated.

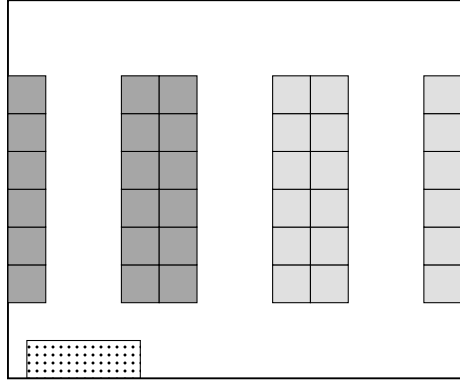
Figure 4.2 depicts four different W-SASs that can be described as follows: $W\text{-SAS}_A$ assigns heavy items to the first half of the warehouse, and light items are stored in the second half of the warehouse. In $W\text{-SAS}_B$, heavy and light items are alternately assigned to picking aisles starting with heavy items in the leftmost picking aisle. In $W\text{-SAS}_C$, heavy items are stored at the respective entrances of the picking aisles, whereas light items are stored within picking aisles. $W\text{-SAS}_D$ stores heavy items below the midpoint of the picking aisle, and light items are stored above.

Table 4.2 shows the performance of $H\text{-PRS}_{W/O}$, $E\text{-PRS}_{W/O}$, and $E\text{-PRS}_W$ for different W-SASs and sorting effort scenarios. Figure 4.3 depicts the average tour lengths of the investigated picker routing strategies for different sorting efforts and different W-SASs.

Comparison of the picker routing strategies without sorting effort If sorting effort is neglected, $E\text{-PRS}_{W/O}$ and $E\text{-PRS}_W$ clearly outperform $H\text{-PRS}_{W/O}$ in the case company on all tested instances with respect to the average total tour length. The average percentage gap to the BKS of $H\text{-PRS}_{W/O}$ is approximately 35%. The comparison of $E\text{-PRS}_{W/O}$ and $E\text{-PRS}_W$ shows that $E\text{-PRS}_W$ deviates between 3.4% and 20.9% from the optimal solutions that are obtained with $E\text{-PRS}_{W/O}$.

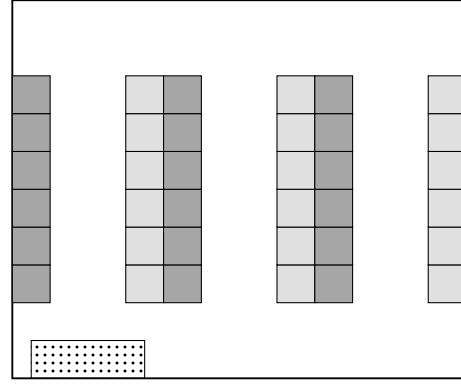
Obviously, $E\text{-PRS}_{W/O}$ is the best performing picker routing strategy. This can be explained by the fact that the sorting of the items takes place after the order picking process and is not considered in the objective function value for $SE = 0$. Interestingly, for $W\text{-SAS}_A$, $E\text{-PRS}_W$ is able to find a near-optimal solution with a deviation of only 3.4% from $E\text{-PRS}_{W/O}$ although for $E\text{-PRS}_{W/O}$ sorting is not considered yet.

Comparison of the picker routing strategies with increasing sorting effort Again, $E\text{-PRS}_{W/O}$ and $E\text{-PRS}_W$ beat the solution quality of $H\text{-PRS}_{W/O}$ on all instances. When comparing the performance of $E\text{-PRS}_{W/O}$ and $E\text{-PRS}_W$, we observe that the superiority of $E\text{-PRS}_W$ in comparison to $E\text{-PRS}_{W/O}$ increases with the sorting effort. For $SE = 3$ and $W\text{-SAS}_A$, $W\text{-SAS}_B$, and $W\text{-SAS}_D$, $E\text{-PRS}_W$ outperforms $E\text{-PRS}_{W/O}$. Recall that a sorting effort of 3 LUs corresponds to 1 second and in-



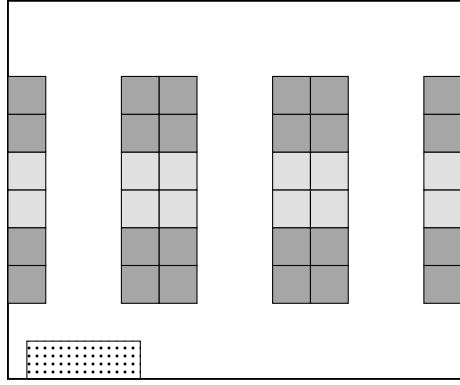
■ A heavy item □ A light item

(a) W-SAS_A: Heavy items are stored in the first half of the warehouse, light items are assigned to the second half of the warehouse.



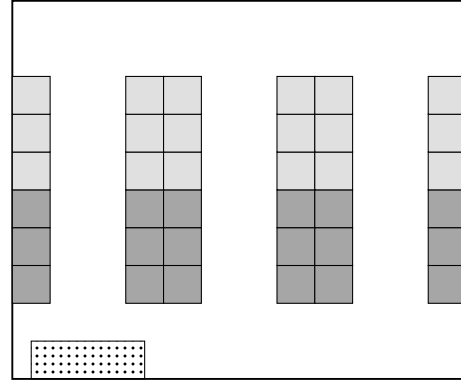
■ A heavy item □ A light item

(b) W-SAS_B: Heavy and light items are alternately stored in the picking aisles, starting with heavy items in the leftmost picking aisle.



■ A heavy item □ A light item

(c) W-SAS_C: Heavy items are stored at the entrances of the picking aisles, light items are stored within the picking aisles.



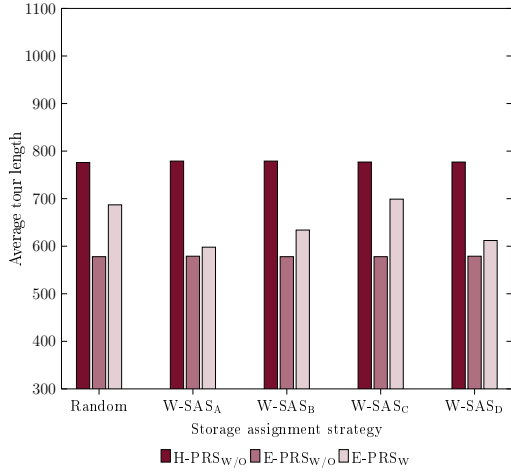
■ A heavy item □ A light item

(d) W-SAS_D: Heavy items are stored below the midpoint of the picking aisle, light items are stored above.

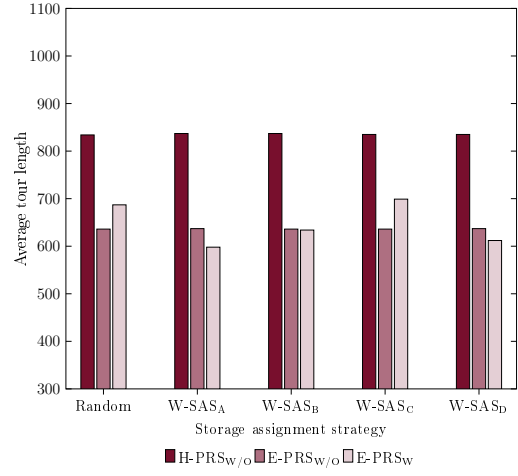
Figure 4.2: Weight-based storage assignment strategies.

W-SAS	SE (LUs)	H-PRS _{W/O}	E-PRS _{W/O}	E-PRS _W
		Δ_f (%)	Δ_f (%)	Δ_f (%)
W-SAS _A	0	34.6	0.0	3.4
W-SAS _A	3	39.9	6.4	0.0
W-SAS _A	6	49.6	16.1	0.0
W-SAS _A	9	59.3	25.8	0.0
W-SAS _A	12	69.0	35.5	0.0
W-SAS _B	0	34.7	0.0	9.6
W-SAS _B	3	32.0	0.4	0.0
W-SAS _B	6	41.1	9.5	0.0
W-SAS _B	9	50.3	18.6	0.0
W-SAS _B	12	59.4	27.8	0.0
W-SAS _C	0	34.4	0.0	20.9
W-SAS _C	3	31.3	0.0	9.8
W-SAS _C	6	28.7	0.0	0.7
W-SAS _C	9	36.1	7.6	0.0
W-SAS _C	12	44.4	15.9	0.0
W-SAS _D	0	34.3	0.0	5.7
W-SAS _D	3	36.4	4.0	0.0
W-SAS _D	6	45.9	13.5	0.0
W-SAS _D	9	55.4	23.0	0.0
W-SAS _D	12	64.8	32.4	0.0

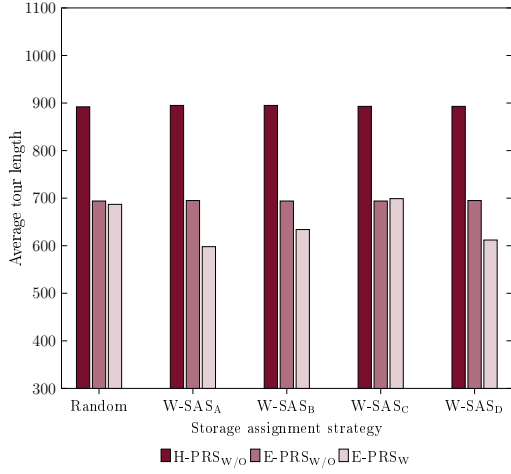
Table 4.2: Performance of the picker routing strategies for different W-SASs. For all comparison strategies, we report the percentage gap between the best solution found by the respective picker routing strategy and the BKS (column Δ_f (%)) for different sorting effort scenarios (column SE (LUs)). The BKS corresponds to the average of the best objective function values obtained for each of the single customer orders by one of the tested picker routing strategies. For each combination of W-SAS and sorting effort scenario, the smallest gap found by any of the picker routing strategies is indicated in bold.



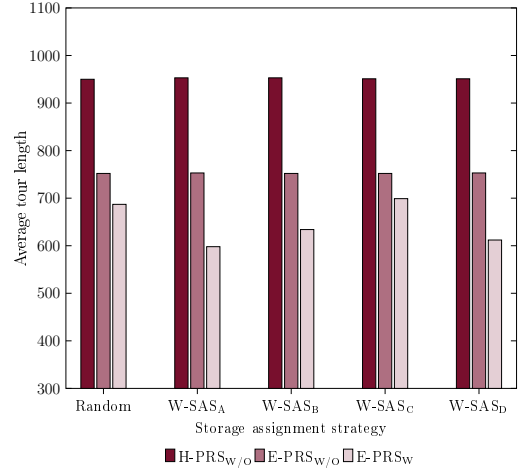
(a) No sorting effort.



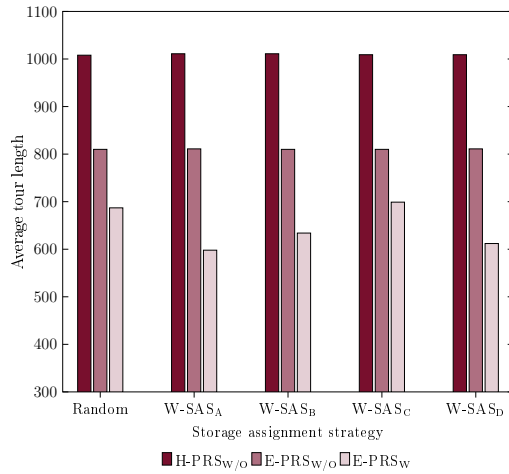
(b) Sorting effort of 3 LUs/requested item.



(c) Sorting effort of 6 LUs/requested item.



(d) Sorting effort of 9 LUs/requested item.



(e) Sorting effort of 12 LUs/requested item.

Figure 4.3: Performance of H-PRSW/O, E-PRSW/O, and E-PRSW for different storage assignments and sorting effort scenarios.

cludes the time for searching an item in a plastic box and the time for identifying an item as “light” or “heavy”. For the practically more realistic sorting effort scenarios ($SE=9,12$), $E\text{-PRS}_{W/O}$ deviates between 7.6% and 35.5% from the BKS that is obtained by $E\text{-PRS}_W$. This indicates a convincing performance of our $E\text{-PRS}_W$.

Effect of different weight-based storage assignment strategies The results that are reported in Table 4.1 and Table 4.2 show that the storage assignment strategies significantly affect the performance of $E\text{-PRS}_W$. In particular, a strong reduction of the average tour length can be achieved by assigning heavy items to the first half of the warehouse and light items to the second half of the warehouse ($W\text{-SAS}_A$). Comparing the results that assume a random storage to those obtained for $W\text{-SAS}_A$ and $SE=0$, the deviation of $E\text{-PRS}_W$ from the BKS is significantly smaller (18.8% versus 3.4%). $W\text{-SAS}_C$ seems not to be appropriate for the given setting because $E\text{-PRS}_W$ deviates by 20.9% from the BKS. $E\text{-PRS}_W$ benefits from a storage assignment strategy according to which heavy items are clearly separated from light items in the warehouse.

To summarize, both the picker routing strategy and the storage assignment strategy have a significant influence on the resulting tour length when addressing the routing of order pickers with the studied precedence constraint. As can be seen from the numerical example, the combination of $E\text{-PRS}_W$ and $W\text{-SAS}_A$ is recommendable for warehouse managers dealing with similar problem settings. Note that it is quite likely that the superiority of $E\text{-PRS}_W$ would increase with further item categories because of the increasing complexity of the sorting process.

4.4 Influence of different problem parameters

In this section, we present numerical studies to analyze the influence of different problem parameters. Because the dataset provided to us by the case company is rather small, we have generated new problem instances to evaluate the influence of different problem parameters on the performance of the picker routing strategies. The newly designed test instances are introduced in Section 4.4.1. Subsequently, we describe the various problem parameters and then examine the influence of these problem parameters on the performance of the picker routing strategies (see Section 4.4.2).

4.4.1 Test instances

Based on the warehouse layout presented in the practical case study, we add more picking aisles to the warehouse to investigate different warehouse sizes, namely 10, 25, and 50 picking aisles. We assign items to storage locations according to W-SAS_A because of its superior performance in the case study.

The instances assume 40 customer orders and a uniformly distributed number of items per customer order, which is randomly drawn from all of the three intervals [5, 35], [36, 70], and [71, 100]. Customer orders vary with respect to the share of heavy and light items: we consider three different mixes with approximately (i) 75% heavy items/25% light items, (ii) 50% heavy items/50% light items, and (iii) 25% heavy items/75% light items per customer order. The carrying capacity of the picking device is sufficient to transport the items of a single customer order on a single order picking tour.

The combination of the described parameter values results in 27 instance classes that are identified by the size of the warehouse, the mix of heavy and light items per customer order, and the number of items per customer order. For each instance class, we generate 20 instances. This leads to $27 \cdot 20 = 540$ instances in total.

4.4.2 Results of the numerical study

In Tables 4.3, 4.4, and 4.5, we aggregate the results of the numerical study. Table 4.3 reports average results for groups of instances defined by the ratio of mixed items (column Ratio of mixed items (%)) and the number of items (column # items) assuming a warehouse with 10 picking aisles, Table 4.4 for 25 picking aisles, and Table 4.5 for 50 picking aisles. Again, we use the average tour length that includes the sorting penalty as a performance measure for comparing the picker routing strategies. For all comparison strategies, we report the percentage gap between the best solution found by the respective picker routing strategy and the BKS (column Δ_f (%)) for different sorting effort scenarios (column SE (LUs)). The BKS corresponds to the average of the best objective function values obtained for each of the individual instances by one of the tested picker routing strategies. The smallest gap found by any of the picker routing strategies is indicated in bold.

Overall comparison of the picker routing strategies The results reported in Tables 4.3, 4.4, and 4.5 show that for all problem parameters, E-PR_{SW/O} and

Ratio of mixed items (%)	# items	SE (LUs)	H-PRSW/O	E-PRSW/O	E-PRSW
			Δ_f (%)	Δ_f (%)	Δ_f (%)
75/25	5-35	0	31.1	0.0	3.9
75/25	5-35	3	36.7	6.7	0.0
75/25	5-35	6	47.2	17.2	0.0
75/25	5-35	9	57.6	27.7	0.0
75/25	5-35	12	68.1	38.2	0.0
75/25	36-70	0	19.9	0.0	1.6
75/25	36-70	3	36.1	16.5	0.0
75/25	36-70	6	54.2	34.6	0.0
75/25	36-70	9	72.2	52.6	0.0
75/25	36-70	12	90.3	70.7	0.0
75/25	71-100	0	13.0	0.0	0.6
75/25	71-100	3	37.5	24.6	0.0
75/25	71-100	6	62.7	49.8	0.0
75/25	71-100	9	87.9	75.0	0.0
75/25	71-100	12	113.1	100.2	0.0
50/50	5-35	0	35.4	0.0	3.5
50/50	5-35	3	40.6	6.3	0.0
50/50	5-35	6	50.3	16.0	0.0
50/50	5-35	9	60.0	25.7	0.0
50/50	5-35	12	69.6	35.4	0.0
50/50	36-70	0	20.4	0.0	2.1
50/50	36-70	3	34.4	14.5	0.0
50/50	36-70	6	50.9	31.0	0.0
50/50	36-70	9	67.4	47.5	0.0
50/50	36-70	12	83.6	64.0	0.0
50/50	71-100	0	9.2	0.0	1.1
50/50	71-100	3	31.3	22.2	0.0
50/50	71-100	6	54.7	45.6	0.0
50/50	71-100	9	78.1	69.0	0.0
50/50	71-100	12	101.4	92.4	0.0
25/75	5-35	0	32.5	0.0	2.6
25/75	5-35	3	39.2	7.5	0.0
25/75	5-35	6	49.3	17.6	0.0
25/75	5-35	9	59.3	27.6	0.0
25/75	5-35	12	69.3	37.6	0.0
25/75	36-70	0	21.3	0.0	2.5
25/75	36-70	3	35.8	15.0	0.0
25/75	36-70	6	53.2	32.4	0.0
25/75	36-70	9	70.7	49.9	0.0
25/75	36-70	12	88.1	67.3	0.0
25/75	71-100	0	13.4	0.0	2.0
25/75	71-100	3	35.9	22.7	0.0
25/75	71-100	6	60.6	47.5	0.0
25/75	71-100	9	85.4	72.2	0.0
25/75	71-100	12	110.1	97.0	0.0

Table 4.3: Performance of the strategies for different problem parameters in a warehouse with 10 picking aisles. The groups of instances are defined by the ratio of mixed items (column Ratio of mixed items (%)) and the number of items (column # items). For all strategies, we report the percentage gap between the best solution found by the respective strategy and the BKS (column Δ_f (%)) for different sorting efforts (column SE (LUs)). The BKS corresponds to the average of the best objective function values obtained for each of the individual instances by one of the tested picker routing strategies. For each instance group and sorting effort, the smallest gap found by any of the strategies is indicated in bold.

Ratio of mixed items (%)	# items	SE (LUs)	H-PRSW/O	E-PRSW/O	E-PRSW
			Δ_f (%)	Δ_f (%)	Δ_f (%)
75/25	5-35	0	33.2	0.0	5.5
75/25	5-35	3	32.4	1.0	0.0
75/25	5-35	6	38.7	7.3	0.0
75/25	5-35	9	45.0	13.5	0.0
75/25	5-35	12	51.2	19.8	0.0
75/25	36-70	0	34.0	0.0	3.9
75/25	36-70	3	39.2	6.6	0.0
75/25	36-70	6	49.6	16.9	0.0
75/25	36-70	9	60.0	27.3	0.0
75/25	36-70	12	70.3	37.7	0.0
75/25	71-100	0	28.2	0.0	2.8
75/25	71-100	3	38.2	10.8	0.0
75/25	71-100	6	51.7	24.3	0.0
75/25	71-100	9	65.1	37.7	0.0
75/25	71-100	12	78.6	51.2	0.0
50/50	5-35	0	35.2	0.0	4.3
50/50	5-35	3	35.6	1.9	0.0
50/50	5-35	6	41.6	7.9	0.0
50/50	5-35	9	47.6	13.9	0.0
50/50	5-35	12	53.7	19.9	0.0
50/50	36-70	0	38.3	0.0	3.9
50/50	36-70	3	43.1	6.2	0.0
50/50	36-70	6	53.1	16.2	0.0
50/50	36-70	9	63.1	26.3	0.0
50/50	36-70	12	73.2	36.3	0.0
50/50	71-100	0	30.4	0.0	3.0
50/50	71-100	3	39.4	9.9	0.0
50/50	71-100	6	52.2	22.6	0.0
50/50	71-100	9	64.9	35.3	0.0
50/50	71-100	12	77.7	48.1	0.0
25/75	5-35	0	32.8	0.0	2.4
25/75	5-35	3	35.8	3.8	0.0
25/75	5-35	6	42.0	10.0	0.0
25/75	5-35	9	48.2	16.2	0.0
25/75	5-35	12	54.4	22.4	0.0
25/75	36-70	0	34.2	0.0	3.3
25/75	36-70	3	40.2	7.1	0.0
25/75	36-70	6	50.4	17.3	0.0
25/75	36-70	9	60.6	27.5	0.0
25/75	36-70	12	70.9	37.7	0.0
25/75	71-100	0	28.0	0.0	3.1
25/75	71-100	3	37.5	10.3	0.0
25/75	71-100	6	50.8	23.7	0.0
25/75	71-100	9	64.2	37.0	0.0
25/75	71-100	12	77.5	50.4	0.0

Table 4.4: Performance of the strategies for different problem parameters in a warehouse with 25 picking aisles. The groups of instances are defined by the ratio of mixed items (column Ratio of mixed items (%)) and the number of items (column # items). For all strategies, we report the percentage gap between the best solution found by the respective strategy and the BKS (column Δ_f (%)) for different sorting efforts (column SE (LUs)). The BKS corresponds to the average of the best objective function values obtained for each of the individual instances by one of the tested picker routing strategies. For each instance group and sorting effort, the smallest gap found by any of the strategies is indicated in bold.

Ratio of mixed items (%)	# items	SE (LUs)	H-PRSW/O	E-PRSW/O	E-PRSW
			Δ_f (%)	Δ_f (%)	Δ_f (%)
75/25	5-35	0	27.4	0.0	5.4
75/25	5-35	3	26.3	0.0	1.0
75/25	5-35	6	29.2	3.1	0.0
75/25	5-35	9	33.3	7.3	0.0
75/25	5-35	12	37.5	11.4	0.0
75/25	36-70	0	37.2	0.0	5.6
75/25	36-70	3	37.0	1.8	0.0
75/25	36-70	6	44.1	8.8	0.0
75/25	36-70	9	51.2	15.9	0.0
75/25	36-70	12	58.3	23.0	0.0
75/25	71-100	0	36.5	0.0	4.7
75/25	71-100	3	39.6	4.7	0.0
75/25	71-100	6	48.8	13.9	0.0
75/25	71-100	9	58.0	23.1	0.0
75/25	71-100	12	67.2	32.3	0.0
50/50	5-35	0	27.4	0.0	3.9
50/50	5-35	3	26.7	0.3	0.0
50/50	5-35	6	30.7	4.3	0.0
50/50	5-35	9	34.7	8.4	0.0
50/50	5-35	12	38.8	12.4	0.0
50/50	36-70	0	40.1	0.0	4.7
50/50	36-70	3	40.8	2.5	0.0
50/50	36-70	6	47.8	9.5	0.0
50/50	36-70	9	54.8	16.5	0.0
50/50	36-70	12	61.8	23.5	0.0
50/50	71-100	0	40.3	0.0	4.4
50/50	71-100	3	43.3	4.7	0.0
50/50	71-100	6	52.2	13.6	0.0
50/50	71-100	9	61.1	22.5	0.0
50/50	71-100	12	70.1	31.5	0.0
25/75	5-35	0	25.3	0.0	1.9
25/75	5-35	3	26.9	2.1	0.0
25/75	5-35	6	30.9	6.0	0.0
25/75	5-35	9	34.8	10.0	0.0
25/75	5-35	12	38.8	14.0	0.0
25/75	36-70	0	37.2	0.0	3.2
25/75	36-70	3	40.2	4.2	0.0
25/75	36-70	6	47.5	11.4	0.0
25/75	36-70	9	54.7	18.6	0.0
25/75	36-70	12	61.9	25.9	0.0
25/75	71-100	0	36.9	0.0	3.4
25/75	71-100	3	41.6	6.0	0.0
25/75	71-100	6	50.8	15.2	0.0
25/75	71-100	9	60.1	24.4	0.0
25/75	71-100	12	69.3	33.6	0.0

Table 4.5: Performance of the strategies for different problem parameters in a warehouse with 50 picking aisles. The groups of instances are defined by the ratio of mixed items (column Ratio of mixed items (%)) and the number of items (column # items). For all strategies, we report the percentage gap between the best solution found by the respective strategy and the BKS (column Δ_f (%)) for different sorting efforts (column SE (LUs)). The BKS corresponds to the average of the best objective function values obtained for each of the individual instances by one of the tested picker routing strategies. For each instance group and sorting effort, the smallest gap found by any of the strategies is indicated in bold.

E-PRSW clearly outperform H-PRSW/O. If no sorting effort is assumed, H-PRSW/O deviates between 9.2% and 40.3% from E-PRSW/O. For increasing sorting efforts, H-PRSW/O has gaps of up to 113.1% from the BKS that is obtained with E-PRSW. When comparing the performance of E-PRSW/O and E-PRSW, we observe that E-PRSW/O slightly outperforms E-PRSW with respect to the average tour length if sorting effort is not considered. Similar to the results of the case study, E-PRSW is able to find near-optimal solutions with a deviation of between 0.6% and 5.6% from E-PRSW/O although for E-PRSW/O sorting is not considered yet. Interestingly, already for a sorting effort of 3 LUs, E-PRSW/O has a deviation of up to 24.6% from E-PRSW. When assuming a sorting effort of 12 LUs, this gap rises to 100.2%.

In the following, we examine how the various problem parameters affect the performance of E-PRSW/O and E-PRSW.

Effect of the warehouse size When comparing the performance of E-PRSW/O and E-PRSW, we observe that E-PRSW/O performs slightly better with increasing warehouse size. Nevertheless, E-PRSW shows a more robust performance with a maximum gap to the BKS of 5.6%, whereas the gaps of E-PRSW/O fluctuate between 0.0% and 100.2%.

Effect of different ratios of mixed items With a higher percentage of light items, the average tour length for the picker routing strategies increases. This is due to the fact that light items are stored in the second half of the warehouse and the order picker therefore has to cover longer travel distances to collect all items of a customer order.

Effect of the number of items per customer order The results reported show that the number of items per customer order significantly affects the performance of all picker routing strategies. E-PRSW performs significantly better if larger customer orders are assumed. For example, E-PRSW shows a gap of only 0.6% to the BKS for the problem setting in which 10 picking aisles, 75% heavy items, 25% light items, [71-100] items per customer orders, and no sorting effort are considered.

4.5 Summary and conclusion

This chapter is inspired by a practical case of a warehouse, in which the item weight influences the sorting sequence of items into cardboard boxes used for shipping the items to the respective customers. The warehouse stores household items, which can be roughly distinguished into heavy (robust) and light (fragile) items. To prevent damage to light items, order pickers collect items of customer orders in plastic boxes without stacking heavy items on top of light items. The route of an order picker through the warehouse for collecting the items of a customer order is determined by a simple S-shape strategy. At the end of the order picking process, the collected items have to be sorted such that the precedence constraint is respected. To avoid the sorting of the collected items after the retrieval process, we propose a picker routing strategy that integrates the precedence constraint by collecting heavy items before light items in an optimal fashion (E-PRSW).

In a case study, we compare E-PRSW to the picker routing strategy applied in the case company (H-PRSW/O) and to an exact solution approach that neglects the precedence constraint (E-PRSW/O). The results show that we improve the current order picking process in the following aspects: Using E-PRSW, warehouse managers are able to completely avoid the sorting of items at the end of the order picking process. Compared to H-PRSW/O, E-PRSW significantly reduces the average travel tour length of an order picker for completing customer orders. With respect to E-PRSW/O, we show that our approach outperforms this strategy if the sorting of an item accounts for two seconds or more.

We also propose different storage assignment strategies considering the weight of items and find that storage assignment significantly affects the performance of the picker routing strategies. The strongest reduction of the average tour length can be achieved by separating heavy and light items in the warehouse and allocating heavy items to storage locations arranged close to the depot. Despite the complexity of the order picking process when order picking is precedence-constrained, our approach is easy to understand for order pickers because it follows a straightforward and non-confusing routing scheme and thus reduces the potential for errors in order picking.

Because the dataset provided to us by the case company is rather small, we generate new problem instances to examine the impact of different problem parameters (warehouse size, share of heavy and light items per customer order, and number of requested items per customer order) on the performance of the picker routing

strategies. We find that E-PRS_W provides near-optimal solutions (even if we compare E-PRS_W to E-PRS_{W/O} that does not consider the precedence constraint) and the most robust solution quality for problem instances with different characteristics.

The intention of our picker routing strategy was to develop an approach that is easy to understand and that can easily be implemented in practice. For handling warehouse operations efficiently, information systems such as warehouse management systems (WMSs) are often used in real-world warehouses. The implementation of our algorithm within a WMS can be easily done. The WMS can deliver all necessary order picking information directly to the order pickers' portable device, such as a radio frequency handheld scanner, a smartphone, or a tablet. Moreover, it is possible to extend the software to feature a graphical user interface visualizing the warehouse layout, the picking locations associated with a customer order, and the route of an order picker through the warehouse.

Chapter 5

Order batching and batch sequencing in an AGV-assisted picker-to-parts system

The contents of this chapter are included in similar form in the following working paper: I. Žulj, H. Salewski, D. Goeke, and M. Schneider. Order batching and batch sequencing in an AGV-assisted picker-to-parts system. Working paper, Deutsche Post Chair – Optimization of Distribution Networks, RWTH Aachen University, 2020.

Efficient order fulfillment processes in warehouses are a key success factor in times of increasing global retail sales volumes and (contractually agreed or promised) next or even same-day deliveries. As described before, reducing the large fraction of unproductive picker walking time of total order picking time is essential for increasing the performance of a traditional picker-to-parts system. To streamline order fulfillment processes, warehouse managers more and more frequently rely on AGV-assisted order picking. By supporting human order pickers with AGVs, the pick density per order picking tour can be increased and unproductive picker walking time can be reduced. For these reasons, this chapter studies the AOPP, which is inspired by a warehouse of a German automotive original equipment manufacturer.

The remainder of this chapter is structured as follows. In Section 5.1, we introduce our AOPP and present a mixed integer program. To solve the problem, we propose a heuristic that combines an ALNS with an adaption of the NEH heuristic in Section 5.2. Section 5.3 is devoted to extensive computational experiments (i) to investigate the effect of the SA-based acceptance criterion used by ALNS/NEH on

the solution quality, (ii) to assess the performance of ALNS/NEH in comparison to CPLEX, and (iii) to derive managerial insights with respect to AGV-assisted order picking. Section 5.4 concludes this chapter with a summary.

5.1 Problem description

In this section, we describe the AOPP and present the mathematical model formulation (AOPP-BE) in which all batches not exceeding the picking device capacity are generated explicitly before solving a given test instance.

Our AOPP considers a rectangular single-block warehouse with parallel closed-end picking aisles of equal length and width that are connected by one orthogonal cross aisle at the front of the picking aisles (see Figure 5.1). Items are stored in storage locations of equal size that are arranged on the left and right side of each picking aisle. Each item is available from exactly one storage location, and each storage location stores exactly one item type. There is a handover location in the cross aisle below the entry of each picking aisle. Let $V = \{1, \dots, n\}$ denote the set of picking aisles. Each handover location is identified through the associated picking aisle $i \in V$. Handover locations and picking aisles, respectively, are numbered in ascending order from the leftmost handover location, represented by 1, to the rightmost handover location, represented by n . The depot is located in the cross aisle below the leftmost handover location. Two instances of the same physical depot are given by 0 and $n + 1$. To indicate that V contains the respective instance of the depot, V is subscripted with 0 or $n + 1$, i.e., $V_0 = V \cup \{0\}$ and $V_{n+1} = V \cup \{n + 1\}$.

In our problem, the items of a set of customer orders O have to be collected from the picking area of the warehouse. A customer order $o \in O$ is specified by a non-empty set of order lines, where each order line indicates a particular item and the requested quantity of this item. Moreover, each customer order $o \in O$ is associated with a due date d_o until which the items contained in the customer order must be collected. Customer orders can be grouped into batches, for which we make the following assumptions:

- *No splitting of customer orders*: The items of a customer order cannot be distributed among different batches because splitting may result in unacceptable sorting effort. Note that by defining the subsets into which customer orders can be split (this may also be single items) as the new customer orders, our AOPP-

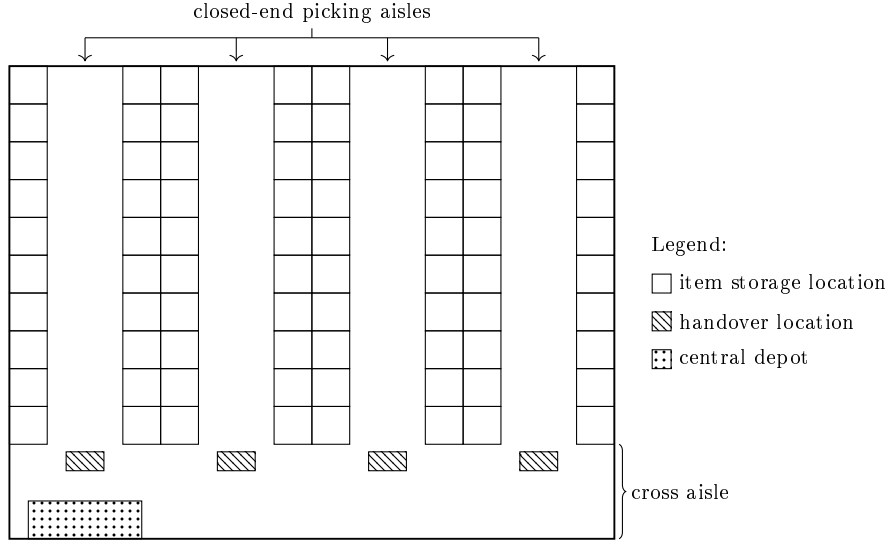


Figure 5.1: Warehouse layout assumed for our AOPP.

BE model could still address the scenario in which splitting of customer orders is allowed.

- *Batch size*: The number of customer orders that may be contained in a batch is restricted by the carrying capacity of the picking device. The capacity is expressed by the number of items which can be carried by the picking device. This has been assumed in other publications as well (see, e.g., Bozer and Kile 2008, Scholz et al. 2017).
- *Generation of batches*: The set of all feasible batches B not exceeding the picking device capacity is generated before solving a test instance. We use binary coefficients v_{bo} to indicate whether customer order $o \in O$ is included in batch $b \in B$ ($v_{bo} = 1$) or not ($v_{bo} = 0$). Because all feasible batches are explicitly defined in the AOPP-BE model, we introduce binary decision variables y_b to specify whether batch b is selected from the set of all feasible batches B ($y_b = 1$) or not ($y_b = 0$). A batch $b \in B$ to which $y_b = 1$ applies is referred to as selected batch in the following.

With respect to the retrieval of the requested items from their storage locations, we make the following decisions and modeling assumptions:

- *Synchronized zone picking*: We consider a synchronized zone picking system (see Section 2.5.5) in which each zone is assigned a single order picker, who is

responsible for retrieving those items of a batch which are stored in the picking aisle of her zone. A zone comprises a single picking aisle and the associated handover location. Obviously, if the items of a batch are stored in different picking aisles, different order pickers are involved in picking these items.

- *Equipment of order pickers:* Each order picker is initially positioned at her handover location and equipped with a picking cart with one or multiple bins as well as a pick list. The bins are used for temporarily storing the retrieved items of a batch, where each bin is dedicated to a single customer order contained in the batch. To illustrate this, consider a batch which consists of customer orders $o=1$ and $o=2$. The requested items of each of these customer orders are stored in picking aisles $i=1$ and $i=2$. Both the picking cart of the order picker assigned to picking aisle $i=1$ and the one assigned to picking aisle $i=2$ is equipped with one bin for the items of customer order $o=1$ and another bin for those of customer order $o=2$.

The pick list of an order picker specifies the selected batches to be processed in her zone and their processing sequence. Moreover, for each of these batches, the pick list indicates (i) the items to be retrieved, (ii) the requested quantities of these items, (iii) the order picker's tour starting from her handover location, proceeding along the storage locations defined by the respective batch, and ending at her handover location, and (iv) the assignment of each of these items to its corresponding bin on the picking cart. In the following, we detail the modeling of the batch processing sequence and the routing of order pickers.

- *Batch processing sequence by order pickers:* The batch processing sequence is described by precedence relationships for all pairs of batches $b, d \in B$. To model the batch processing sequence, we introduce binary decision variables z_{bd} , which denote whether the order pickers process batch $b \in B$ before batch $d \in B$ ($z_{bd} = 1$) or not ($z_{bd} = 0$).
- *Routing of order pickers:* The order picker is guided by the following routing scheme when picking the batch items stored in her picking aisle: Starting from the handover location, she enters the picking aisle and walks to the farthest storage location in which a requested item is stored. Here, she retrieves the requested number of items and places them in the associated bins. All other picking locations in this picking aisle are successively visited on the way back to the handover location. Given the above described setting of the order picking

system with closed-end picking aisles and a single cross aisle, an order picker is optimally routed (concerning the objective of minimizing travel time) in this way.

To model the retrieval sequence, we introduce the sets V^b , V_0^b , V_{n+1}^b , and $V_{0,n+1}^b$, the continuous decision variables s_{bi} , and the parameters p_{bi} . V^b denotes the picking aisles from which the items of batch $b \in B$ have to be retrieved (called relevant picking aisles of batch $b \in B$). We subscript V^b with 0 and/or $n+1$, i.e., $V_0^b = V^b \cup \{0\}$, $V_{n+1}^b = V^b \cup \{n+1\}$, and $V_{0,n+1}^b = V^b \cup \{0\} \cup \{n+1\}$ to indicate that V^b contains the respective instance of the depot. Variables s_{bi} describe the order picker's start time at handover location $i \in V^b$ for picking the batch items stored in picking aisle $i \in V^b$. Parameters p_{bi} denote the picking time comprising (i) the time the order picker spends on walking from her handover location $i \in V^b$ to the farthest picking location and from the last visited picking location back to handover location $i \in V^b$, (ii) the time she requires to walk between the picking locations of batch $b \in B$ in picking aisle $i \in V^b$, and (iii) the time for retrieving the requested items from their storage locations and placing them into the appropriate bins on the picking cart.

Upon collection of the batch items stored in picking aisle $i \in V^b$, the order picker returns to her handover location. Contrary to standard order picking approaches in which an order picker transports the picked items to a depot, we assume that the order picker remains at her handover location, where she passes the items to an AGV.

The AGV-assistance in the considered order picking system is based on the following decisions and modeling assumptions:

- *AGV fleet*: A fleet of m identical AGVs is initially located at the depot.
- *Equipment of AGVs*: Recall that exactly one AGV collects the picked items of a single batch from the relevant handover locations (i.e., those from which the items of a batch are to be collected) on a single tour through the warehouse. To this end, an AGV is equipped with one or multiple bins for temporarily storing the batch items, where each of the bins is intended for the items of a single customer order contained in a batch. We assume that (i) all requested items of a customer order fit into a single bin, and (ii) the AGV capacity is sufficient to carry all items of a single batch.

- *AGV tours*: An AGV tour starts from the depot, proceeds along the cross aisle to the relevant handover locations, and ends at the depot. On an AGV tour, each of the handover locations of a batch is visited exactly once. Because of safety reasons, an AGV may only change the travel direction along the cross aisle at the rightmost handover location from which batch items are to be collected, and all other relevant handover locations have to be visited on the way to this handover location. Passing of other AGVs is allowed.

To model an AGV tour, we use the following notation. Binary decision variables x_{bij} indicate whether an AGV collecting the items of batch $b \in B$ drives from depot/handover location $i \in V_0^b$ to handover location/depot $j \in V_{n+1}^b$ ($x_{bij} = 1$) or not ($x_{bij} = 0$). Parameters t_{ij} denote the travel time from depot/handover location $i \in V_0$ to handover location/depot $j \in V_{n+1}$, and we assume that the triangle inequality applies, i.e., $t_{ij} \leq t_{ik} + t_{kj}$, where $k \in V$. Moreover, travel times are assumed to be deterministic and symmetric, i.e., $t_{ij} = t_{ji}$. Continuous decision variables a_{bi} denote the arrival time of the AGV collecting the items of batch $b \in B$ at handover location $i \in V^b$.

- *Assignment of batches to AGVs and batch processing sequence by AGVs*: In the AOPP-BE, we also decide on the assignment of batches to AGVs and the sequence according to which the batches assigned to an AGV are to be processed by the AGV. Both decisions are implicitly modeled by defining precedence relationships for all pairs of batches $b, d \in B$. To this end, we define that a batch has either a direct predecessor batch or no direct predecessor batch if it is the first batch processed by a certain AGV. To model the batch processing sequence, we introduce binary decision variables ζ_{bd} and α_b . Variables ζ_{bd} denote whether the items of batch $b \in B$ are collected directly before those of batch $d \in B$ by a certain AGV ($\zeta_{bd} = 1$) or not ($\zeta_{bd} = 0$). Variables α_b state whether batch $b \in B$ is processed first by a certain AGV ($\alpha_b = 1$) or not ($\alpha_b = 0$).

As described above, an order picker returns to her handover location $i \in V^b$ after picking the batch items stored in picking aisle $i \in V^b$. Here, she immediately places the picked items into the appropriate bins of the AGV handling batch $b \in B$ (called associated AGV) if the AGV has already arrived at her handover location $i \in V^b$, or she waits until it arrives. To model handovers, we use the following notation. Continuous decision variables h_{bi} denote the start time of handing over the items of batch $b \in B$ at handover location $i \in V^b$ to the associated AGV. The associated

times for the handovers are given by parameters l_{bi} .

After an order picker has handed over the items to the AGV, she equips the picking cart with (new) bins, and she returns to her picking aisle to process the next batch according to the pick list. The time she spends on equipping the picking cart with (new) bins is assumed to be negligible.

The AGV moves either towards other relevant handover locations, or it returns to the depot if all items of the batch that is currently handled by the AGV are collected. Upon arrival at the depot, it takes u_b time units to unload the items of batch $b \in B$. The batch is then considered as completed, and the AGV is equipped with (new) bins for collecting the items of the next batch. Continuous decision variables f_b describe the completion time of batch $b \in B$.

We measure the quality of a solution to the AOPP-BE in terms of the total tardiness of all customer orders because customer orders often have to be completed until given due dates to avoid delays in shipments to customers or in production. The total tardiness is computed as $\tau = \sum_{o \in O} \tau_o$, where continuous decision variables τ_o define the tardiness of customer order $o \in O$ as the positive difference between the time the customer order is completed and its due date. Note that the completion time of a customer order corresponds to the completion time of the batch in which the customer order is contained.

To summarize, our AOPP-BE models the following decisions such that the total tardiness of all customer orders is minimized:

- Which customer orders are to be grouped together to form a batch?
- Which of the selected batches should be processed by which AGV?
- In which sequence should the selected batches be processed by the order pickers and by the AGVs?

To provide a compact overview of the AGV-assisted order picking process described above, Figure 5.2 summarizes the process from the perspective of an order picker and Figure 5.3 from the perspective of an AGV.

Finally, Figure 5.4 demonstrates precedence relationships between two order pickers and a single AGV when processing a single batch in the considered system. Consider a batch which consists of items that are stored in both picking aisles of the given warehouse. The order picker assigned to the left picking aisle starts from her handover location at time 0 and returns with the picked items from her picking

aisle at time 10. The other order picker starts at time 0 and returns at time 4. The associated AGV starts from the depot at time 0 and first proceeds along the cross aisle to the left handover location, which it reaches at time 2. As soon as the respective order picker arrives at her handover location, she immediately starts handing over the picked items to the AGV already waiting at the handover location. After the handover is completed at time 12, the AGV drives to the right handover location, which it reaches at time 14. The order picker assigned to the right picking aisle arrives at her handover location before the AGV and waits 10 time units for the AGV. Upon arrival of the AGV, she hands over the picked items to the AGV. Finally, the AGV returns to the depot and reaches it at time 19. Here, the unloading of the collected items immediately starts and finishes at time 22.

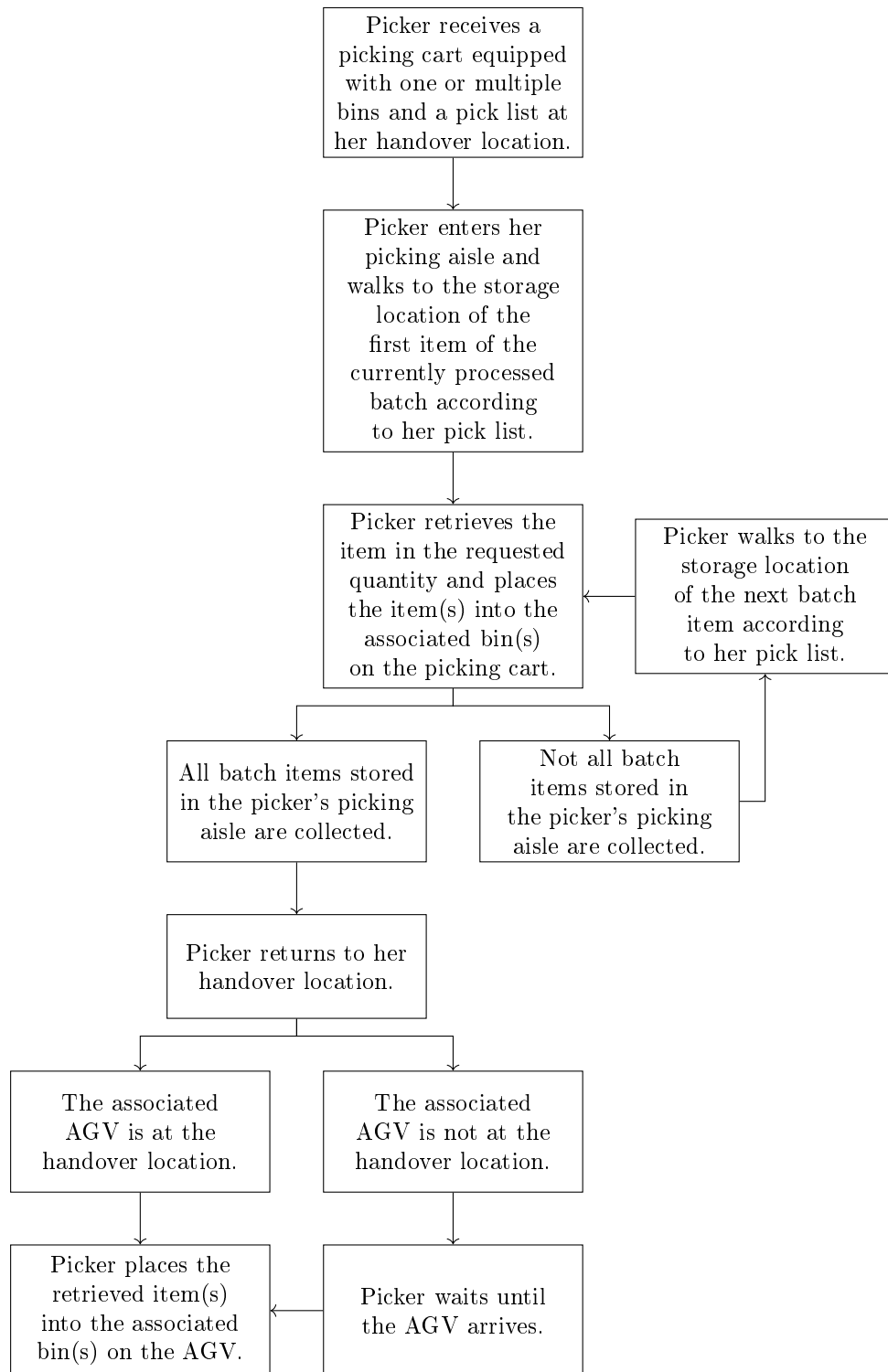


Figure 5.2: Order picking process from the perspective of an order picker in the AGV-assisted order picking system.

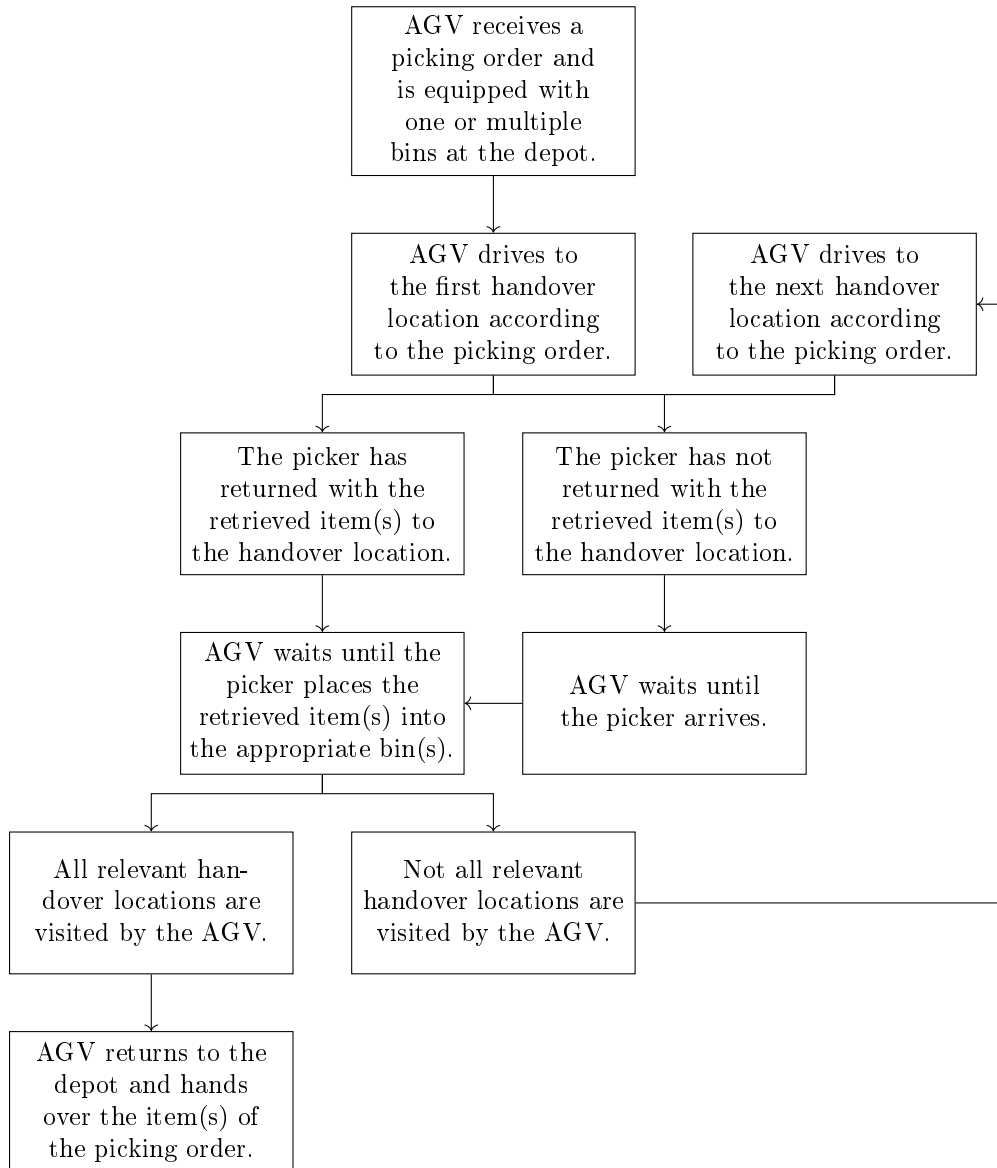


Figure 5.3: Order picking process from the perspective of an AGV in the AGV-assisted order picking system.

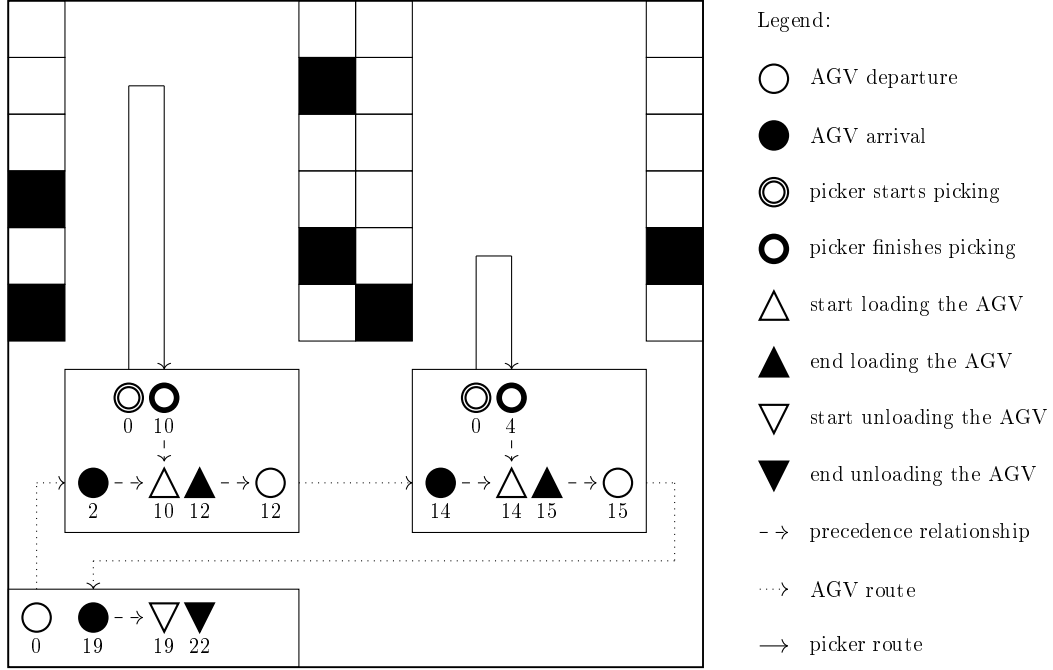


Figure 5.4: An example illustrating precedence relationships between two order pickers and a single AGV when processing a single batch. The black rectangles represent picking locations defined by the batch.

In the following, we present the mathematical model formulation of our AOPP-BE. Note that M denotes a sufficiently large positive number, which must be at least as large as the maximum completion time over all batches. However, because the completion time of a batch is not known before solving a specific test instance, we calculate M based on the worst case scenario according to which a batching of customer orders is not allowed, and customer orders are sequentially picked, i.e., it is not possible to process several customer orders simultaneously. Then, M can be calculated as follows:

$$M = \sum_{o \in O} \left(\max_{i \in V} \{p'_{oi}\} + \sum_{i \in V} l'_{oi} + u'_o + 2 \cdot \max_{i, j \in V_{0, n+1}} \{t_{ij}\} \right) \quad (5.1)$$

Parameters p'_{oi} denote the time the order picker requires to pick those items of customer order $o \in O$ which are stored in her picking aisle $i \in V$, parameters l'_{oi} indicate the time the order picker requires to pass the items of customer order $o \in O$ at handover location $i \in V$ to the associated AGV, and parameters u'_o specify the time required to unload the items of customer order $o \in O$ from the associated AGV. As introduced above, parameters t_{ij} represent the travel time from depot/handover

location $i \in V_0$ to handover location/depot $j \in V_{n+1}$.

Using the notation summarized in Table 5.1, our AOPP-BE can be formulated as a mixed integer problem consisting of objective function (5.2) and constraints (5.3) to (5.18).

$0, n+1$	depot instances
Sets	
B	set of feasible batches (indices: b, d)
O	set of customer orders (index: o)
V^b	set of relevant picking aisles of batch $b \in B$, where each relevant handover location $i \in V^b$ is identified through the associated picking aisle $i \in V^b$ (indices: i, j)
V_0^b	set of depot instance 0 and relevant picking aisles of batch $b \in B$, where $V_0^b = V^b \cup \{0\}$ (indices: i, j)
V_{n+1}^b	set of depot instance $n+1$ and relevant picking aisles of batch $b \in B$, where $V_{n+1}^b = V^b \cup \{n+1\}$ (indices: i, j)
$V_{0,n+1}^b$	set of depot instances and relevant picking aisles of batch $b \in B$, where $V_{0,n+1}^b = V^b \cup \{0\} \cup \{n+1\}$ (indices: i, j)
Parameters	
d_o	due date of customer order $o \in O$
l_{bi}	time the order picker requires to pass the items of batch $b \in B$ at handover location $i \in V^b$ to the associated AGV
m	number of AGVs
M	a sufficiently large positive number
p_{bi}	time the order picker requires to pick the items of batch $b \in B$ which are stored in picking aisle $i \in V^b$
t_{ij}	time an AGV requires to travel from depot/handover location $i \in V_0$ to depot/handover location $j \in V_{n+1}$
u_b	time required to unload the items of batch $b \in B$ from the associated AGV
v_{bo}	1, if customer order $o \in O$ is included in batch $b \in B$; 0, otherwise
Continuous decision variables	
a_{bi}	arrival time of the AGV handling batch $b \in B$ at handover location $i \in V^b$
f_b	completion time of batch $b \in B$
h_{bi}	order picker's start time of passing the items of batch $b \in B$ to the associated AGV at handover location $i \in V^b$
s_{bi}	order picker's start time of picking the items of batch $b \in B$ at handover location $i \in V^b$
τ_o	tardiness of customer order $o \in O$
Binary decision variables	
α_b	1, if batch $b \in B$ is the first batch handled by a certain AGV; 0, otherwise
x_{bij}	1, if the AGV handling batch $b \in B$ travels from depot/handover location $i \in V_0^b$ to depot/handover location $j \in V_{n+1}^b$; 0, otherwise
y_b	1, if batch $b \in B$ is selected from the set of feasible batches B ; 0, otherwise
z_{bd}	1, if batch $b \in B$ is handled before batch $d \in B$ by the order pickers; 0, otherwise
ζ_{bd}	1, if batch $b \in B$ is handled directly before batch $d \in B$ by a certain AGV; 0, otherwise

Table 5.1: Overview of the notation used in the AOPP-BE model.

$$\text{minimize } \sum_{o \in O} \tau_o \quad (5.2)$$

subject to

$$\sum_{b \in B} y_b \cdot v_{bo} = 1 \quad \forall o \in O \quad (5.3)$$

$$\sum_{\substack{j \in V_{n+1}^b \\ j > i}} x_{bij} = y_b \quad \forall i \in V_0^b; b \in B \quad (5.4)$$

$$\sum_{\substack{i \in V_0^b \\ i < j}} x_{bij} = y_b \quad \forall j \in V_{n+1}^b; b \in B \quad (5.5)$$

$$z_{bd} + z_{db} \geq 1 - M \cdot (2 - y_b - y_d) \quad \forall b, d \in B; b \neq d \quad (5.6)$$

$$\alpha_d + \sum_{\substack{b \in B \\ b \neq d}} \zeta_{bd} \geq y_d \quad \forall d \in B \quad (5.7)$$

$$\sum_{\substack{d \in B \\ d \neq b}} \zeta_{bd} \leq y_b \quad \forall b \in B \quad (5.8)$$

$$\sum_{b \in B} \alpha_b \leq m \quad (5.9)$$

$$h_{bi} + l_{bi} - M \cdot (1 - z_{bd}) \leq s_{di} \quad \forall b, d \in B; b \neq d; i \in V^b \cup V^d \quad (5.10)$$

$$s_{bi} + p_{bi} \leq h_{bi} \quad \forall b \in B; i \in V^b \quad (5.11)$$

$$a_{bi} \leq h_{bi} \quad \forall b \in B; i \in V^b \quad (5.12)$$

$$h_{bi} + l_{bi} + t_{ij} - M \cdot (1 - x_{bij}) \leq a_{bj} \quad \forall b \in B; i \in V_0^b; j \in V^b; i \neq j \quad (5.13)$$

$$f_b + t_{0,i} - M \cdot (2 - x_{d,0,i} - \zeta_{bd}) \leq a_{di} \quad \forall b, d \in B; b \neq d; i \in V^d \quad (5.14)$$

$$h_{bi} + l_{bi} + t_{i,n+1} + u_b - M \cdot (1 - x_{b,i,n+1}) \leq f_b \quad \forall b \in B; i \in V^b \quad (5.15)$$

$$f_b - d_o - M \cdot (1 - v_{bo} \cdot y_b) \leq \tau_o \quad \forall b \in B; o \in O \quad (5.16)$$

$$a_{bi}, f_b, h_{bi}, s_{bi}, \tau_o \geq 0 \quad \forall b \in B; i \in V_{0,n+1}^b; o \in O \quad (5.17)$$

$$\alpha_b, x_{bij}, y_b, z_{bd}, \zeta_{bd} \in \{0, 1\} \quad \forall b, d \in B; i, j \in V_{0,n+1}^b \quad (5.18)$$

The objective of minimizing the total tardiness of all customer orders is defined in (5.2). Constraints (5.3) guarantee that each customer order is included in exactly one of the selected batches. Constraints (5.4) and (5.5) state that an AGV collecting the items of a batch starts from the depot, drives to the relevant handover locations from the leftmost to the rightmost, and then returns to the depot. It is also assured

that the relevant handover locations are visited exactly once on each AGV tour.

Constraints (5.6) define the sequence according to which the batches are to be processed by the order pickers. The sequence in which the batches are handled by the AGVs is modeled in constraints (5.7), (5.8), and (5.9). Constraints (5.7) enforce that each selected batch is either the first batch or direct predecessor batch of another batch (or multiple batches) handled by the same AGV. Constraints (5.8) assure that each selected batch has at most one direct successor batch. Constraints (5.9) state that at most one batch can be the first batch handled by a single AGV.

Constraints (5.10) define the order picker's start time of picking the batch items stored in her picking aisle. If a batch $d \in B$ is not the first batch processed by the order picker, we link the order picker's start time of picking the items of batch $d \in B$ to the time at which the order picker has handed over the items of the direct predecessor batch $b \in B$ to the associated AGV. Constraints (5.11) and (5.12) guarantee that the order picker cannot pass the picked items to the associated AGV until (i) she has returned to the handover location with these items (see constraints (5.11)), and (ii) the AGV has arrived at her handover location (see constraints (5.12)). Constraints (5.13) determine the arrival time of the AGV handling batch $b \in B$ at each relevant handover location $i \in V^b$. Constraints (5.14) link the arrival time of the AGV handling batch $d \in B$ at relevant handover location $i \in V^d$ to the completion time of batch $b \in B$ (plus the time the AGV requires to travel from the depot to relevant handover location $i \in V^d$) if the following holds: (i) batch $b \in B$ is handled by the same AGV as batch $d \in B$, and (ii) batch $b \in B$ is handled before batch $d \in B$ by the AGV. Constraints (5.15) compute the completion time for each batch $b \in B$, and constraints (5.16) calculate the tardiness for each customer order $o \in O$. Finally, the continuous decision variables and the binary decision variables are defined in constraints (5.17) and (5.18), respectively.

An advantage of the mathematical model presented above is that it is easy to understand. However, it requires that all feasible batches are generated before solving a specific test instance. Consequently, the number of variables and constraints in the model depends on the number of feasible batches, which increases exponentially with the number of customer orders. In Appendix A, we present our AOPP-BI model, in which the batches are not generated in advance.

5.2 Solution approach

In this section, we present our two-stage solution approach consisting of an ALNS component and an adaption of the well-known NEH algorithm to solve our AOPP. Section 5.2.1 details different metrics used to efficiently evaluate changes to a (partial) solution of our problem. In Section 5.2.2, we introduce the ALNS component for grouping customer orders into batches. Subsequently, the generated batches are sequenced by using an NEH-based heuristic (see Section 5.2.3). Note that an optimal (i.e., minimum total tardiness) assignment of batches to AGVs is achieved by assigning the batches to the next available AGV while respecting the given batch processing sequence.

5.2.1 Metrics used in the solution approach

The evaluation of the true objective function of our problem is time-consuming. To illustrate this, we use the example presented in Figure 5.4 (see Section 5.1). We assume that the batch consists of two customer orders, namely $o=1$ and $o=2$. The items of customer order $o=1$ are stored in the left picking aisle and those of customer order $o=2$ in both picking aisles. Let us consider a destroy operator which removes customer order $o=1$ from the given batch and thus causes the following changes: The time the order picker spends on retrieving the items from the left picking aisle is reduced by two time units, and the time to hand over these items to the associated AGV decreases by one time unit (compared to the scenario given in Figure 5.4). The order picker reaches the handover location at time 8, and she immediately starts handing over the items. After the handover is completed, the AGV drives to the right handover location and arrives there at time 11. Accordingly, the start and end time of the subsequent order picking activities change (except the retrieval of the items from the right picking aisle). Even this small example shows that a slight modification of the solution requires many recalculations to evaluate the true objective function. Obviously, such recalculations would increase in the case of, e.g., multiple batches affected by the modification.

In the testing phase of our algorithm, we identified different metrics, which are correlated to the true objective function but are less time-consuming to evaluate. The following metrics are used to evaluate the moves performed by, e.g., the repair and destroy operators in the ALNS:

- ξ_b denotes the surrogate completion time of batch b , which is computed in the same way as the completion time f_b (see Section 5.1), except that for the computation of ξ_b , b is considered to be the only batch that has to be processed by the the order pickers and the associated AGV.
- φ_b corresponds to the surrogate tardiness of batch b . We compute φ_b as follows:

$$\varphi_b = \sum_{o \in O_b} \max \{0; \xi_b - d_o\} \quad (5.19)$$

O_b denotes the set of customer orders contained in batch b (recall that the due date of customer order o is given by d_o).

- η_s specifies the surrogate total tardiness of all batches $b \in B_s$ in batching solution s , which is computed as follows:

$$\eta_s = \sum_{b \in B_s} \varphi_b \quad (5.20)$$

5.2.2 Adaptive large neighborhood search for batching customer orders

This section details our ALNS component for grouping customer orders into batches. In Figure 5.5, a pseudocode overview of the ALNS component is given. First, we generate an initial batching solution s by using an adaption of the C&W(ii) algorithm (see Section 5.2.2.1). Next, s is improved by several ALNS iterations, in which neighboring batching solutions are accepted according to an SA-based acceptance criterion (see Section 5.2.2.2).

5.2.2.1 Initial solution

We generate an initial batching solution by implementing an adaption of the C&W(ii) algorithm. To minimize the total tardiness of all customer orders, in problem settings with loose due dates, it may be reasonable to sort customer orders in descending order of their due date and then group them until the capacity of an AGV is no longer sufficient (see, e.g., Scholz et al. 2017). However, in problem settings with tight due dates, not only the due dates of the customer orders should be considered when generating batches. In particular, generating batches with short completion times positively affects the minimization of the total tardiness of all customer orders

```

generate an initial batching solution  $s$  using an adaption of the C&W(ii) algorithm
 $s^* \leftarrow s$ 
repeat
  randomly draw  $q^-$  customer orders to be removed
  choose destroy and repair operators  $(h_i^-, h_i^+)$  according to probabilities  $(\pi_i^-, \pi_i^+)$ 
   $s' \leftarrow h_i^+(h_i^-(s))$ 
  if acceptSA( $s'$ ) then
     $s \leftarrow s'$ 
  end if
  if  $\eta_{s'} < \eta_{s^*}$  then
     $s^* \leftarrow s'$ 
  end if
  adjust the weights  $w_i$  and probabilities  $\pi_i$  of the destroy and repair operators
until stop criterion is met
return  $s^*$ 

```

Figure 5.5: Overview of the ALNS algorithm.

(see Section 2.4).

Therefore, we group customer orders into batches dependent on (i) the similarity of their due date and (ii) the saving in terms of the surrogate completion time reduction, which results from processing customer orders simultaneously instead of separately. Our procedure for generating an initial batching solution can be described as follows:

- Each customer order in the considered problem instance is initially assigned to a single batch.
- We compute relatedness measure ϕ_{bd} for each pair of batches b and d for which the AGV carrying capacity is sufficient as follows:

$$\phi_{bd} = \frac{\xi_b + \xi_d - \xi_{bd}}{\max\{\varepsilon; \max_{b \in B} \{\xi_b\} - \min_{b \in B} \{\xi_b\}\}} \cdot \frac{\min\{\delta_b; \delta_d\}}{\max\{\varepsilon; \max_{b \in B} \{\delta_b; \delta_d\}\}}, \quad (5.21)$$

where ξ_{bd} denotes the surrogate completion time when processing batches b and d simultaneously, parameters δ_b (δ_d) represent the minimum due date among all customer orders which are contained in batch b (d), and ε specifies a small positive number. Consequently, the term $\xi_b + \xi_d - \xi_{bd}$ describes the surrogate completion time saving that results from picking both batches b and d simultaneously instead of separately.

We normalize the surrogate completion time saving using the extreme values

of the surrogate completion times across the set of all batches, which are given by the problem instance, and we weight them with the quotient of $\min \{\delta_b; \delta_d\}$ and $\max \{\varepsilon; \max \{\delta_b; \delta_d\}\}$ to stimulate the grouping of batches with similar due dates.

- Subsequently, the pairs of batches are sorted in non-ascending order of ϕ_{bd} . Our algorithm starts with the pair of batches with the largest surrogate completion time saving. The following three cases may occur while grouping a pair of batches (b, d) to a larger batch: (i) a new batch is generated if neither of the two batches b and d is yet assigned, (ii) if one of the batches b and d is already assigned, the other batch is assigned to the same batch, provided that the AGV carrying capacity is sufficient, (iii) the next pair of batches is considered if both of the batches b and d are already assigned to larger batches or if the AGV carrying capacity is not sufficient. Note that we recalculate the surrogate completion time savings ϕ_{bd} after each grouping of batches to larger batches.

Subsequently, we aim at improving the resulting initial batching solution by means of an ALNS (see Section 5.2.2.2).

5.2.2.2 Adaptive large neighborhood search component

LNS iteratively destroys and subsequently repairs solutions by removing and reinserting a relatively large number of customer orders. As described in Section 2.6.4, ALNS is an extension of LNS using multiple destroy and repair operators within the same search process, which are chosen based on the performance of the destroy and repair operators in past iterations. The probability with which an operator is chosen is dynamically modified during the search process.

Destroy and repair operators As input, our three destroy operators take a batching solution s and an integer q^- , where q^- denotes the number of customer orders to be removed. The output of the operators is a partial solution in which q^- customer orders have been removed. To remove customer orders from the current batching solution, ALNS uses the following destroy operators:

Random removal randomly removes customer orders from the current batching solution until q^- customer orders are removed.

Worst removal aims at removing customer orders which appear to be unfavorably assigned to batches in the current batching solution s with respect to the surrogate total tardiness η_s of batching solution s .

To this end, we compute for each customer order the reduction of the surrogate total tardiness η_s when removing customer order o from the current batching solution s . Then, we sort all customer orders in a list of size L in descending order of the surrogate total tardiness reduction. At each iteration, we choose the customer order at position $\lfloor L \cdot u^p \rfloor$ in this list, where u denotes a uniformly distributed number in $[0, 1)$ and p a randomization parameter in order to avoid repeatedly removing the same customer orders. We repeat this procedure until q^- customer orders are removed.

Relatedness removal proposed by Shaw (1997) removes customer orders that are related to each other according to several criteria and thus likely to be easily interchangeable. If related customer orders are removed, there are more possibilities for reinsertion and thus new, potentially better batching solutions can be found. Otherwise, if customer orders are removed that are very different from each other, they will probably be reinserted into their original batches due to unattractive reinsertion possibilities.

To define the relatedness between two customer orders, we use the relatedness measure introduced in Section 5.2.2.1. Recall that a high relatedness measure value indicates that two customer orders are highly related to each other. The first customer order to be removed is randomly chosen. All remaining customer orders are sorted in a list in descending order of their relatedness to the selected customer order. Then, we choose the customer order at position $\lfloor L \cdot u^p \rfloor$ in this list of size L , where again u denotes a uniformly distributed number in $[0, 1)$ and $p \geq 1$ a randomization parameter, which allows to balance the influence of randomness and relatedness on the selection of the customer orders to be removed. In the case of $p = 1$, customer orders are removed at random, and if $p = \infty$, customer orders are selected based on their respective relatedness measure value. Afterwards, we randomly choose a customer order from those already removed from the batching solution, and we repeat the described procedure until q^- customer orders are removed.

To reinsert the previously removed customer orders into the current batching solution, ALNS uses one of the following three repair operators:

Random insertion aims at solution diversification. To this end, random insertion randomly selects a batch and inserts the customer order into it.

Greedy insertion iteratively assigns customer orders to the batch with minimal increase in surrogate total tardiness η_s of batching solution s .

Regret insertion Note that in the last iterations of the ALNS, greedy insertion tends to insert those customer orders whose insertion leads to a large increase in the surrogate total tardiness η_s of batching solution s . A drawback of a later insertion is that there are not many (attractive) possibilities to insert the customer orders because a relatively large number of customer orders is already contained in the batches (recall that the batch size is limited). Regret insertion was proposed by Ropke and Pisinger (2006b) and improves the greedy insertion by anticipating the future effect of an insertion operation.

The k -regret value of each customer order o describes the change in the value of the surrogate total tardiness η_s of inserting the customer order in its k -best batch and its best batch. Obviously, k describes the extent to which the future is anticipated. The customer order with the largest absolute k -regret value is selected for insertion, and the procedure is repeated until all customer orders are inserted. We implement the 2-regret and 3-regret heuristic. In preliminary studies, we found that larger values of k did not improve the solution quality.

Adaptive mechanism For the sake of convenience, we briefly repeat the procedure of adaptive mechanism introduced in Section 2.6.4. Adaptive weight adjustment evaluates the importance of each destroy and repair operator by modifying the probability with which an operator is chosen based on the performance of the operator in past iterations. All operators are set to the same initial weight ω . Performance of an operator $i \in \mathcal{X}$ is measured by the score value o_i . If an operator finds a new best batching solution, we increase the score by o_{best} , if a better batching solution is found by o_{imp} , and if a new deteriorating batching solution is found and accepted according to the SA-based acceptance criterion by o_{acc} . We divide the search process of the ALNS component into a number of segments of γ iterations. After γ iterations, the new weight of operator $i \in \mathcal{X}$ is calculated as $w_{i+1} = (1 - r) \cdot w_i + r \cdot \frac{o_i}{\beta_i}$, where w_i corresponds to the weight of operator $i \in \mathcal{X}$. The reaction parameter $r \in [0, 1]$ controls the speed of the weight adjustment and takes the success of an operator in previous segments into account. The number of times that an operator $i \in \mathcal{X}$ was

chosen in the previous segment is denoted by β_i . Afterwards, the probability π_i to choose an operator $i \in \mathcal{X}$ is calculated according to $\pi_i = w_i / \sum_{i \in \mathcal{X}} w_i$.

Acceptance criterion We use an acceptance criterion based on SA in order to overcome local optima. An improving solution is always accepted, and a new deteriorating solution s' is accepted with probability $e^{-(\eta_{s'} - \eta_s)/t_i}$, where $t_i > 0$ is the current temperature. The starting temperature t_0 is determined such that a solution that deteriorates the current solution by $a\%$ is accepted with a probability of 50%. We decrease the temperature by a constant factor c each time a deteriorating solution is accepted such that in the last 20% of iterations the temperature is below 0.0001.

5.2.3 Construction heuristic for deciding the batch processing sequence

To determine the batch processing sequence, we propose an adaption of the NEH algorithm introduced by Nawaz et al. (1983). The idea of our batch sequencing algorithm is that a batch b which has a larger surrogate tardiness φ_b should be given higher priority to be processed before another batch d with less surrogate tardiness φ_d . In Figure 5.6, we give an overview of the proposed algorithm, which is described in the following:

- First, we calculate surrogate tardiness φ_b for each batch $b \in B_s$ contained in the batching solution s .
- Second, we determine sequence $\lambda := (\lambda_1, \dots, \lambda_\kappa)$, in which the batches are arranged by decreasing surrogate tardiness φ_b , e.g., λ_1 denotes the batch with largest surrogate tardiness, and λ_2 indicates the batch with the second largest surrogate tardiness.
- Third, we select the batch with the largest surrogate tardiness, i.e., λ_1 , and we insert it in the partial batch processing sequence π^* , i.e., $\pi^* := (\lambda_1)$.
- Last, for $i = 2$ to κ , we do the following: we insert batch λ_i in the position of π^* (among the i possible ones) which minimizes the (partial) total tardiness τ . The partial batch processing sequence with the smallest (partial) total tardiness τ defines the relative positions of these i batches with respect to each other.

```

for each batch  $b \in B_s$  contained in the batching solution  $s$ , calculate its surrogate tardiness  $\varphi_b$ 
generate sequence  $\lambda := (\lambda_1, \dots, \lambda_\kappa)$  by sorting the batches in non-ascending order of their surrogate tardiness  $\varphi_b$ 
 $\pi^* := (\lambda_1)$ 
for  $i \leftarrow 2$  to  $\kappa$  do
    insert batch  $\lambda_i$  in the position of  $\pi^*$  which minimizes the (partial) total tardiness  $\tau$ 
end for
return  $\pi^*$ 

```

Figure 5.6: Overview of the NEH-based algorithm.

5.3 Numerical studies

This section describes the numerical studies (i) to investigate the influence of the SA-based acceptance criterion used by ALNS/NEH on the solution quality, (ii) to assess the performance of ALNS/NEH in comparison to CPLEX, and (iii) to give managerial insights with respect to AGV-assisted order picking. Because no established testbed is available for our problem, we generate three new sets of instances, which we describe in Section 5.3.1. Subsequently, the parameter setting of our ALNS (see Section 5.3.2) and the results of our experiments are presented in Section 5.3.3, Section 5.3.4, and Section 5.3.5.

5.3.1 Instance generation

We generate test instances of two different sizes. The set of small instances is denoted by S , and sets M and L comprise large-sized instances. The instance generation process is described in the following:

- *Warehouse layout:* We consider three different sizes of single-block warehouses with parallel closed-end picking aisles (see Section 5.1), namely $n = 10$, $n = 15$, and $n = 20$ picking aisles. The warehouse which comprises 10 picking aisles contains 20 storage locations in each picking aisle, 10 on the left and 10 on the right. The warehouse containing 15 picking aisles consists of 30 storage locations per picking aisle (15 on each side of the picking aisle), and in the warehouse with 20 picking aisles, 40 storage locations are arranged in each picking aisle (20 on each side of the picking aisle). Thus, 200 different items are stored in the warehouse with $n = 10$ picking aisles, 450 in the warehouse with $n = 15$ picking aisles, and 800 in the warehouse with $n = 20$ picking aisles.

The warehouses apply a random storage assignment strategy according to which items are randomly assigned to storage locations. The depot is located below the leftmost handover location. The instances in set S are based on the warehouse with $n = 10$ picking aisles, those in set M , on $n = 10$, $n = 15$, or $n = 20$ picking aisles, and those in set L , on $n = 15$ or $n = 20$ picking aisles.

- *Physical dimensions of the warehouses:* An order picker has to cover 1.5 LUs to get from her handover location to the first storage location in the associated picking aisle. The distance between two adjacent storage locations amounts to 1 LU. An AGV moves 1 LU from the depot to the leftmost handover location. The travel distance between two neighboring handover locations is 5 LUs.
- *Time parameters for order picking activities:* The time an order picker requires to retrieve an item from its storage location is 8 seconds, and the time an order picker spends on handing over an item to an AGV amounts to 4 seconds. It takes 4 seconds to unload an item from an AGV at the depot. Due to safety reasons, the traveling speed of AGVs is restricted to the walking speed of order pickers: both move 1 LU in 3 seconds.
- *Number and capacity of picking devices:* The number of AGVs is fixed to $m = 1$ and $m = 2$ in set S , to $m = 4$ in set M , and to $m = 6$ in set L . The carrying capacity C of both AGV and picking cart is $C = 6$ or $C = 8$ items in set S and $C = 60$ items in sets M and L .
- *Generation of pick lists:* The number of customer orders is $|O| = 5, 6, 7$ in set S , $|O| = 40, 60, 80, 100$ in set M , and $|O| = 40, 50, 60, 70, 80, 90, 100$ in set L . In the small-sized instances, the number of items contained in a customer order is uniformly distributed over the interval $[2, 4]$. Note that in e-commerce warehouses, the average number of requested items per customer order is approximately two items (see Boysen et al. (2019b)) so that this setting is realistic for such warehouses. In sets M and L , the number of items is randomly drawn from the uniformly distributed interval $[5, 25]$.
- *Generation of due dates:* To generate due dates, we assign to each customer order contained in a problem instance a due date of 0 and solve the resulting AOPP-BI model using ALNS/NEH. The minimum and maximum completion time over the resulting batches define the extreme values of the uniformly distributed interval of due dates, from which we randomly choose a due date for each customer order in the respective problem instance.

Combining the above described parameter values leads to 12 instance groups for set S , to 12 instance groups for set M , and to 14 instance groups for set L . The instance groups associated with set S are identified by the number of customer orders $|O|$, the number of AGVs m , and the capacity C . The groups of large-sized instances are identified by the number of picking aisles n and the number of customer orders $|O|$. For each instance group, we generate 10 instances, i.e., $12 \cdot 10 = 120$ small-sized instances, $12 \cdot 10 = 120$ instances for set M , and $14 \cdot 10 = 140$ instances for set L .

5.3.2 Parameter setting for the adaptive large neighborhood search component

For tuning the parameters of our ALNS component, we follow the approach described in Ropke and Pisinger (2006a). We start with the parameter setting used for the ALNS component of ALNS \times TS for the standard OBP. Then, we change the value of a single parameter while keeping the rest of the parameters fixed and perform five runs on randomly selected instances from the sets of instances described in Section 5.3.1. We choose the parameter setting with the best average result as the final setting for the respective parameter, and we repeat this procedure with the next parameter. We randomly determine the order in which the tuning of the parameters is performed. The ALNS component seems relatively insensitive to the variation of parameters because none of the tested parameter settings results in significant deterioration of solution quality. We found the following parameter setting, which is used for all computational studies.

The scores used in the adaptive weight adjustment are set to $o_{best} = 120$, $o_{imp} = 100$, and $o_{acc} = 80$. The initial weight of all operators amounts to $\omega = 10$. We fix the number of iterations after which the weights of the ALNS component are adjusted to $\gamma = 100$, the reaction factor to $\epsilon = 0.4$, the randomization parameter to $p = 3$, and the percentage of deterioration to $a = 25\%$. The number of customer orders removed from the current solution q^- is randomly drawn from the interval $[q_{min} = 0.175 \cdot |O|, q_{max} = 0.35 \cdot |O|]$, where again $|O|$ denotes the number of customer orders.

To achieve a good trade-off between the solution quality and the runtime of our algorithm, we set the number of ALNS iterations to $|O|$. Table 5.2 summarizes the parameter setting of the ALNS component.

All experiments are executed on a desktop computer with an Intel Core i7-3770 Processor at 3.5 GHz, 16 GB of memory, and Windows 7 Professional. The solution

method is implemented using Java, and the IBM ILOG CPLEX solver (version 12.9) is applied to solve the mixed integer program presented in Appendix A.

Parameter	Parameter value
γ	100
q_{min}, q_{max}	$0.175 \cdot O , 0.35 \cdot O $
ϵ	0.4
$o_{best}, o_{imp}, o_{acc}$	120, 100, 80
ω	10
p	3
a	25%

Table 5.2: Overview of the parameter setting of the ALNS component used to generate a batching solution.

5.3.3 Effect of the simulated annealing-based acceptance criterion

This section analyzes the effect of using an SA-based acceptance criterion after the ALNS phase instead of an ALNS/NEH heuristic accepting only improving solutions (ALNS/NEH w/o SA). In Table 5.3, we give an overview of the results obtained on the large-sized instances in set L and present averages for groups of instances defined by the number of picking aisles (column n) and the number of customer orders (column $|O|$). The reported results are based on five runs. In column BKS , we provide the BKS as the average of the best objective function values obtained by one of the tested methods for each of the individual instances in the group over the five runs. For ALNS/NEH and ALNS/NEH w/o SA, we give the following information:

- $\Delta_f(\%)$ denotes the percentage gap between the best solution found by the respective variant and the BKS over the five runs. The percentage gap is computed as $100 \cdot (f_k - BKS) / f_k$, where f_k denotes the best solution achieved by variant $k \in K$. The smallest gap found by any of the variants is indicated in bold.
- $t(s)$ reports the average runtime over the five runs in seconds.

The results show that ALNS/NEH outperforms ALNS/NEH w/o SA with an average gap of the total tardiness of 0.5% to the BKS compared to an average gap of 1.04% obtained by ALNS/NEH w/o SA. Thus, using the SA-based acceptance criterion instead of simply accepting improving solutions leads to a larger reduction of the average total tardiness of all customer orders. With respect to runtimes, the table shows that both variants require similar runtimes on average.

n	$ O $	BKS	ALNS/NEH		ALNS/NEH w/o SA	
			Δ_f (%)	t (s)	Δ_f (%)	t (s)
15	40	5572	0.76	1.6	0.89	1.5
15	50	8594	0.26	3.3	0.81	3.0
15	60	10214	0.53	6.3	1.77	6.9
15	70	13649	0.33	13.8	0.85	12.3
15	80	20442	0.48	21.2	1.58	23.6
15	90	22819	0.85	35.0	1.13	25.3
15	100	27502	0.70	47.9	1.28	46.9
20	40	5648	0.40	1.7	0.85	1.7
20	50	7607	0.40	3.4	0.98	3.7
20	60	9571	0.17	6.5	1.34	6.7
20	70	15430	0.47	13.2	0.90	13.9
20	80	20655	0.35	28.4	0.75	24.5
20	90	29111	0.63	37.4	0.64	33.7
20	100	27658	0.68	52.8	0.73	51.4
Average			0.50	19.5	1.04	18.2

Table 5.3: Effect of using an SA-based acceptance criterion on the large-sized instances in set L . In the first two columns, we specify the group of instances defined by the number of picking aisles (column n) and the number of customer orders (column $|O|$). In column BKS , we provide the BKS as the average of the best objective function values obtained by one of the tested methods for each of the individual instances in the group. For ALNS/NEH and ALNS/NEH w/o SA, the table reports the percentage gap between the best solution found by the respective variant and the BKS over the five runs (column Δ_f (%)) and the average runtime over the five runs in seconds (column t (s)). The smallest gap found by any of the variants is indicated in bold.

5.3.4 Comparison of the solution method to a commercial solver

This section assesses the performance of ALNS/NEH on the small-sized instances. We compare the performance of CPLEX solving AOPP-BI (see Appendix A) and ALNS/NEH. We perform five runs of ALNS/NEH on each instance, and we restrict

the solution time limit of CPLEX solving AOPP-BI to 1800 seconds.

Table 5.4 presents aggregate results on the small-sized instances and reports averages for groups of instances defined by the number of customer orders (column $|O|$), the number of AGVs (column m), and the carrying capacity of the picking device (column C). In column *BKS*, we provide the BKS as the average of the best objective function values obtained with the respective method for each of the individual instances in the group. For CPLEX and ALNS/NEH, we give the following information:

- $\#opt$ indicates the number of instances solved to optimality. The optimality of a solution is proven by CPLEX.
- $\Delta_f(\%)$ denotes the percentage gap between the best solution found by the respective method and the BKS. The percentage gap is computed as $100 \cdot (f_k - BKS)/f_k$, where f_k denotes the best solution achieved by method $k \in K$.
- $t(s)$ reports the average runtime in seconds.

In Appendix B, we provide detailed computational results of the comparison methods for each of the small-sized instances.

Although the number of customer orders ranges only between 5 and 7 customer orders, each consisting of 2, 3, or 4 items to be picked, CPLEX is not able to consistently solve the instances to optimality within the given runtime limit. The table shows that CPLEX guarantees an optimal solution for 61 of the 120 instances. On all tested instances, ALNS/NEH is able to match the solution quality of CPLEX in five of five runs per instance. Concerning runtime, CPLEX requires significantly more runtime (975.4 seconds on average) than ALNS/NEH (0.019 seconds on average).

The results of ALNS/NEH clearly demonstrate its ability to address our problem on small-sized instances.

5.3.5 Experiments on the AGV fleet size and speed

This section describes the design of our experiments and presents their results. In the experiments, we investigate the effect of increasing (i) the number of AGVs and (ii) the AGV speed on the average total tardiness. We focus on these two problem parameters because they can be easily adjusted by warehouse managers without extensively redesigning the order picking system. All experiments are based on the instances of set M (see Section 5.3.1).

$ O $	m	C	BKS	CPLEX			ALNS/NEH		
				#opt	Δ_f (%)	t (s)	#opt	Δ_f (%)	t (s)
5	1	6	196	10	0.0	61.9	10	0.0	0.027
5	1	8	175	10	0.0	56.9	10	0.0	0.016
5	2	6	137	10	0.0	14.0	10	0.0	0.009
5	2	8	144	10	0.0	17.7	10	0.0	0.010
6	1	6	788	0	0.0	1800.0	0	0.0	0.021
6	1	8	229	1	0.0	1620.9	1	0.0	0.014
6	2	6	133	8	0.0	982.9	8	0.0	0.014
6	2	8	121	8	0.0	669.6	8	0.0	0.024
7	1	6	287	1	0.0	1620.3	1	0.0	0.029
7	1	8	295	0	0.0	1800.0	0	0.0	0.014
7	2	6	268	3	0.0	1261.2	3	0.0	0.019
7	2	8	314	0	0.0	1800.0	0	0.0	0.027
Average					0.0	975.4		0.0	0.019
Minimum					0.0	14.0		0.0	0.009
Maximum					0.0	1800.0		0.0	0.029

Table 5.4: Performance of ALNS/NEH in comparison to CPLEX on small-sized instances in set S . In the first three columns, we specify the group of instances defined by the number of customer orders (column $|O|$), the number of AGVs (column m), and the carrying capacity of the picking device (column C). In column BKS , we provide the BKS as the average of the best objective function values obtained with the respective method for each of the individual instances in the group. For CPLEX and ALNS/NEH, the table reports the number of instances solved to optimality (column #opt), the percentage gap between the best solution found by the respective method and the BKS (column Δ_f (%)) and the average runtime in seconds (column t (s)).

We consider three different sizes of the AGV fleet, i.e., $m=4$, $m=8$, and $m=12$ AGVs, and we vary the speed ratio σ from 1.0 to 3.0 in steps of 0.5, where σ defines the quotient of traveling speed of AGVs and walking speed of order pickers. For each combination of m and σ , we compute the average of the best objective function values obtained by ALNS/NEH for each of the individual instances in the respective instance group over five runs. Recall that each instance group is defined by the warehouse size n and the number of customer orders $|O|$. The resulting values are compared to the scenario (called reference case) with the same warehouse size n and the same number of customer orders $|O|$ but with the AGV fleet size fixed to $m=4$ and the AGV speed restricted to the walking speed of order pickers, i.e., $\sigma=1.0$.

For each instance group and pair of m and σ , Table 5.5 reports the change in

average total tardiness in percent compared to the average total tardiness of the respective reference case. Column m denotes the number of AGVs, and the remaining columns are divided into three blocks, where each block reports the results for one of the three warehouse sizes for different speed ratios (columns σ). Additionally, in Figures 5.7, 5.8, and 5.9, we illustrate the average total tardiness for all combinations of different numbers of AGVs and speed ratios for the three warehouse sizes. Figure 5.7 shows the results for $n=10$ picking aisles, Figure 5.8 for $n=15$ picking aisles, and Figure 5.9 for $n=20$ picking aisles.

In the following, we discuss the results of our experiments:

Effect of the AGV speed The results show that the AGV speed significantly influences the average total tardiness. For example, for $n=10$ picking aisles and $m=4$ AGVs, a marginal increase in the speed ratio from $\sigma=1.0$ to $\sigma=1.5$ reduces the average total tardiness over the different numbers of customer orders by 49.5%. This indicates that the AGV travel times account for a substantial share of the time required to complete the customer orders (i.e., order picking time).

As can be expected, the strongest reductions are achieved at the largest speed ratio of $\sigma=3.0$: for example, for $n=15$ picking aisles, $m=4$ AGVs, and a speed ratio of $\sigma=1.5$, the average total tardiness over the different numbers of customer orders decreases by 63.4%, while at $\sigma=3.0$, the reduction amounts to 88.2%.

However, the results show that with increasing speed ratio, the additional reduction of the average total tardiness tends to decline. For example, for $n=20$ picking aisles and $m=4$ AGVs, an increase in the speed ratio from $\sigma=1.0$ to $\sigma=1.5$ reduces the average total tardiness over the different numbers of customer orders by 76.4%, from $\sigma=1.0$ to $\sigma=2.0$ by 90.5%, from $\sigma=1.0$ to $\sigma=2.5$ by 93.8%, and from $\sigma=1.0$ to $\sigma=3.0$ by 95.1%. Obviously, even at low speed ratios, e.g., $\sigma=1.5$, a large number of customer orders is completed until the respective due date. Consequently, there is not much room left for reducing the average total tardiness. So, by further increasing the AGV speed, the additional reductions obtained are not that large.

We make the following observations when comparing the results obtained for different warehouse sizes and numbers of customer orders:

- *Warehouse size*: The larger the warehouse, the more distances may have to be covered by both the order pickers and the AGVs to complete a given set of customer orders. Therefore, an increase in the speed ratio has the strongest

effect in the case of the largest warehouse. For example, on the instances with $n = 10$ picking aisles and $m = 4$ AGVs, an increase in the speed ratio from $\sigma = 1.0$ to $\sigma = 1.5$ reduces the average total tardiness over the different numbers of customer orders by 49.5%, while on the instances which assume $n = 20$ picking aisles, the reduction amounts to 76.4%.

- *Number of customer orders:* When comparing the results obtained for a small number of customer orders to those for a large number of customer orders, we observe that a slight increase in the AGV speed tends to have a larger impact on the average total tardiness assuming a small number of customer orders. For example, for $n = 15$ picking aisles, $|O| = 40$ customer orders, and $m = 4$ AGVs, an increase in the speed ratio from $\sigma = 1.0$ to $\sigma = 1.5$ reduces the average total tardiness by 68.3%, while for $|O| = 100$ customer orders, a reduction of 55.3% is achieved. To understand this rather non-intuitive result, we examined the solutions obtained on the respective instances in more detail and found the following: On the instances with a small number of customer orders, the share of the AGV travel times of the order picking time (and thus the impact on the average total tardiness) is significantly larger compared to other components of the order picking time (e.g., waiting time of AGVs at handover locations). As a result, increasing the AGV speed leads to a relatively large reduction of the average total tardiness. However, with an increasing number of customer orders, the impact of AGV travel times weakens, while the impact of the other components increases. Thus, in the case of a large number of customer orders, an increase in the AGV speed has only a smaller effect on reducing the average total tardiness.

Effect of the number of AGVs The larger the number of AGVs, the larger the reduction of the average total tardiness. However, the reported values show that expanding the AGV fleet from $m = 4$ to $m = 8$ AGVs leads to larger reductions than from $m = 8$ to $m = 12$ AGVs: for example, for the largest warehouse and a speed ratio of $\sigma = 1.0$, Table 5.5 reports that a doubling of the AGV fleet size from $m = 4$ to $m = 8$ reduces the average total tardiness over the different numbers of customer orders by 84.6%, while by further increasing the AGV fleet size (from $m = 8$ to $m = 12$), an additional reduction of only 0.6 percentage points is achieved.

We observe the following when comparing the results over the different warehouse sizes and numbers of customer orders:

- *Warehouse size*: In the largest warehouse, an increase in the AGV fleet size has a stronger effect on the average total tardiness compared to the results which are obtained in the case of the smallest warehouse. Again, this might be due to the large distances that have to be covered by the AGVs in the largest warehouse. For example, in the smallest warehouse and for a speed ratio of $\sigma = 1.0$, an increase in the AGV fleet size from $m = 4$ to $m = 8$ reduces the average total tardiness over the different numbers of customer orders by 53.3%. In the largest warehouse, the reduction amounts to 84.6%.
- *Number of customer orders*: In the case of a small number of customer orders, an expansion of the AGV fleet from $m = 4$ to $m = 8$ offers less potential for reduction in comparison to the results obtained on the instances with a large number of customer orders. For example, on the instances with $n = 10$ picking aisles, $|O| = 40$ customer orders, and a speed ratio of $\sigma = 1.0$, a doubling of the number of AGVs from $m = 4$ to $m = 8$ reduces the average total tardiness by 49.6%, while on those with $|O| = 100$ customer orders, a reduction of 60.1% is realized. We have examined this in more detail and found that for smaller numbers of customer orders, there are already enough AGVs to achieve a good solution quality so that expanding the AGV fleet size hardly affects the average total tardiness.

Overall managerial insights An interesting result is observed on the instances assuming $|O| = 40$ customer orders: In all warehouses, a marginal increase in the speed ratio of 0.5 is more advantageous than doubling the number of AGVs from $m = 4$ to $m = 8$. For example, for $n = 10$ picking aisles (and $|O| = 40$ customer orders), an increase in the speed ratio from $\sigma = 1.0$ to $\sigma = 1.5$ leads to a reduction of 55.5% of the average total tardiness, while expanding the AGV fleet from $m = 4$ to $m = 8$ (and fixing the speed ratio to $\sigma = 1.0$) reduces the average total tardiness by 49.6%. In the case of $m = 8$ AGVs, this observation does not only hold true for $|O| = 40$ customer orders. Here, in almost all cases, an increase in the speed ratio of 0.5 results in a larger reduction of the average total tardiness compared to increasing the number of AGVs from $m = 8$ to $m = 12$ AGVs. For example, in the warehouse with $n = 15$ picking aisles, increasing the speed ratio from $\sigma = 1.0$ to $\sigma = 1.5$ (assuming $m = 8$ AGVs) reduces the average total tardiness over all customer orders by 84.0%, while an expansion of the AGV fleet from $m = 8$ to $m = 12$ results in a reduction of 73.8%.

The results suggest that a fleet size of $m=8$ AGVs and a speed ratio of $\sigma=1.5$ seem to be a reasonable combination in our computational experiments. In real-world warehouses, the traveling speed of AGVs is restricted to or is only slightly faster than the walking speed of order pickers due to safety reasons (see, e.g., Löffler et al. 2020). Thus, speed ratios of $\sigma=1.0$ and $\sigma=1.5$ are the norm. The fact that speed ratios of $\sigma \geq 2.0$ only contribute to minor additional reductions compared to a speed ratio of $\sigma=1.5$ is therefore also a managerial reason to aim for a speed ratio of $\sigma=1.5$ (when dealing with similar problem settings).

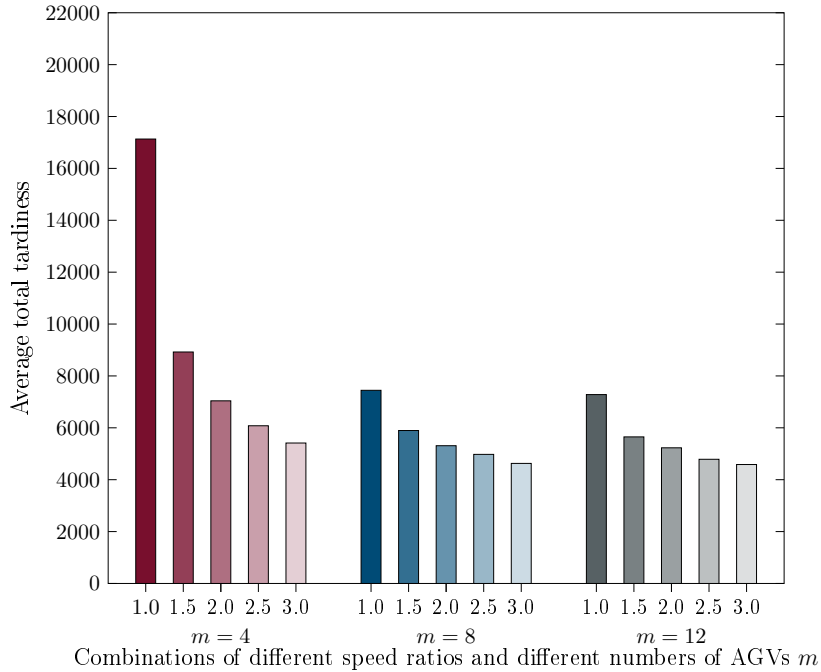


Figure 5.7: Average total tardiness that results on the instances assuming a warehouse with 10 picking aisles. We depict the average total tardiness for all combinations of different speed ratios and numbers of AGVs.

5.4 Summary and conclusion

In this chapter, we investigate a novel warehousing concept using AGVs to support human order pickers of the traditional picker-to-parts setup. Our AOPP decides on the grouping of customer orders into batches, the assignment of these batches to AGVs, and the sequence according to which these batches are to be processed by the order pickers and the AGVs such that the total tardiness of all customer orders is minimized.

O	m	n = 10					n = 15					n = 20				
		$\sigma = 1.0$	$\sigma = 1.5$	$\sigma = 2.0$	$\sigma = 2.5$	$\sigma = 3.0$	$\sigma = 1.0$	$\sigma = 1.5$	$\sigma = 2.0$	$\sigma = 2.5$	$\sigma = 3.0$	$\sigma = 1.0$	$\sigma = 1.5$	$\sigma = 2.0$	$\sigma = 2.5$	$\sigma = 3.0$
40	4	0.0	-55.5	-74.6	-80.9	-81.5	0.0	-68.3	-85.2	-92.8	-93.1	0.0	-82.2	-94.5	-98.2	-98.6
40	8	-49.6	-69.1	-77.9	-81.2	-81.9	-62.6	-81.0	-89.9	-93.1	-94.2	-76.1	-91.7	-97.6	-98.2	-98.6
40	12	-51.2	-71.0	-78.5	-81.8	-82.7	-62.9	-81.4	-90.0	-93.2	-94.7	-76.4	-94.0	-97.6	-98.5	-98.8
60	4	0.0	-46.7	-59.0	-66.1	-74.4	0.0	-66.2	-81.2	-89.2	-90.2	0.0	-82.3	-93.8	-95.2	-96.9
60	8	-46.6	-63.0	-71.3	-71.8	-75.2	-75.7	-86.4	-92.5	-92.6	-93.1	-86.0	-92.0	-94.4	-96.7	-97.9
60	12	-48.2	-63.9	-71.5	-72.1	-75.3	-75.9	-88.1	-92.6	-92.7	-93.6	-86.7	-93.8	-96.1	-97.0	-98.1
80	4	0.0	-49.5	-65.2	-70.5	-70.6	0.0	-63.7	-81.7	-83.9	-88.9	0.0	-72.7	-87.3	-90.2	-93.2
80	8	-56.8	-71.9	-72.7	-75.8	-77.1	-76.1	-86.0	-88.7	-89.9	-90.7	-86.9	-92.7	-94.4	-94.6	-94.7
80	12	-57.7	-72.7	-73.9	-76.0	-77.2	-78.9	-86.1	-89.3	-90.2	-90.9	-87.5	-92.8	-94.4	-94.7	-95.3
100	4	0.0	-46.4	-53.2	-58.5	-63.7	0.0	-55.3	-69.2	-78.3	-80.6	0.0	-68.5	-86.6	-91.5	-91.9
100	8	-60.1	-62.2	-65.0	-66.6	-68.8	-76.5	-82.5	-83.9	-84.5	-86.1	-89.6	-93.9	-94.0	-95.1	-95.3
100	12	-60.8	-64.1	-65.2	-68.5	-69.1	-77.4	-83.4	-84.1	-85.0	-86.7	-89.8	-95.1	-95.2	-95.4	-95.5
Average	4	0.0	-49.5	-63.0	-69.0	-72.6	0.0	-63.4	-79.3	-86.0	-88.2	0.0	-76.4	-90.5	-93.8	-95.1
Average	8	-53.3	-66.5	-71.7	-73.8	-75.8	-72.7	-84.0	-88.8	-90.0	-91.0	-84.6	-92.6	-95.1	-96.1	-96.6
Average	12	-54.5	-67.9	-72.3	-74.6	-76.1	-73.8	-84.7	-89.0	-90.2	-91.4	-85.1	-93.9	-95.8	-96.4	-96.9

Table 5.5: Change in objective function values (in percent). For each instance group (defined by the number of customer orders $|O|$ and the warehouse size n) and pair of m and σ , we report the change in average total tardiness in percent compared to the average total tardiness of the respective reference case. A reference case is identified by the warehouse size n , the number of customer orders $|O|$, $m = 4$ AGVs, and $\sigma = 1.0$, and it is indicated in bold.

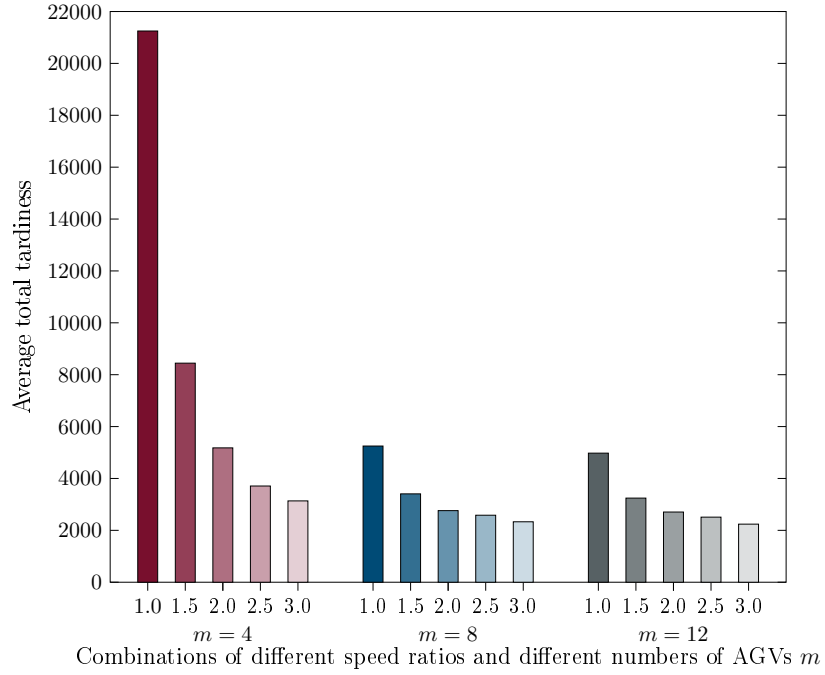


Figure 5.8: Average total tardiness that results on the instances assuming a warehouse with 15 picking aisles. We depict the average total tardiness for all combinations of different speed ratios and numbers of AGVs.

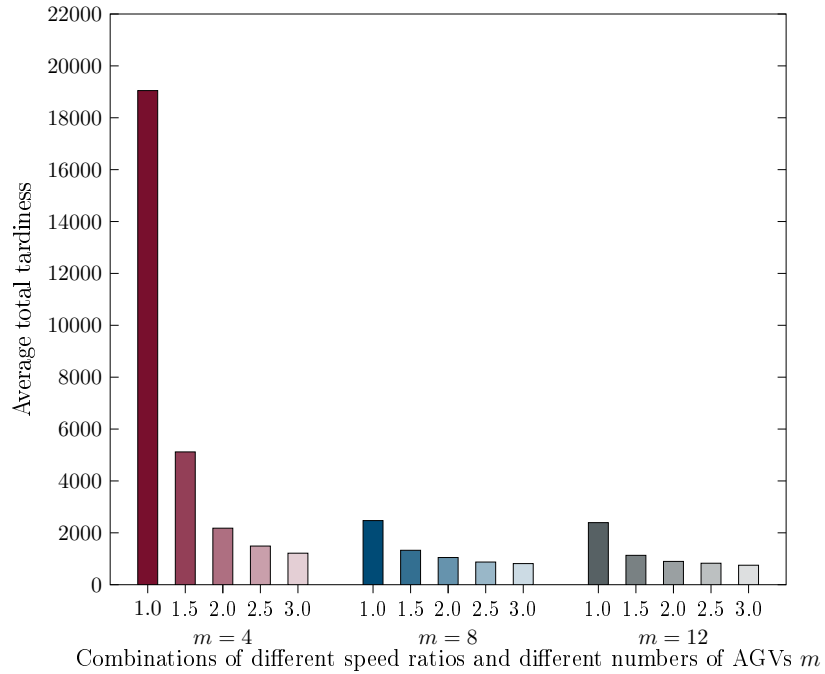


Figure 5.9: Average total tardiness that results on the instances assuming a warehouse with 20 picking aisles. We depict the average total tardiness for all combinations of different speed ratios and numbers of AGVs.

To solve the NP-hard problem, we develop a two-stage heuristic solution approach consisting of an ALNS and an adaption of the well-known NEH heuristic. We use the ALNS component for generating batches, and we implement the NEH-based heuristic to determine the sequence according to which these batches are to be processed.

In numerical studies performed on newly designed instances, we first show the positive effect of our SA-based acceptance criterion after the ALNS phase compared to an ALNS/NEH heuristic that accepts only improving solutions. Subsequently, we demonstrate the strong performance of ALNS/NEH to solve small-sized instances. More precisely, ALNS/NEH is able to match the solution quality of CPLEX on small-sized instances within a fraction of a second.

Finally, we give managerial insights into the benefits of increasing (i) the number of AGVs and (ii) the AGV travel speed on the average total tardiness. In our experiments, we find that by adding (or removing) AGVs or by increasing (or decreasing) the AGV speed to adapt to different workloads, a large number of customer orders can be completed until the respective due date. Moreover, in most of the tested cases, a slight increase in the speed ratio leads to a larger reduction of the average total tardiness compared to an expansion of the AGV fleet. Finally, the results suggest a speed ratio of $\sigma = 1.5$. This finding might be also interesting for warehouse managers: for safety reasons, the AGV speed is only allowed to be slightly faster than the walking speed of order pickers in real-world warehouses.

Chapter 6

Summary, conclusion, and future research

Order picking has been identified as a crucial factor for the competitiveness of a supply chain because inadequate order picking performance causes customer dissatisfaction and high costs. This dissertation aims at designing new models and algorithms to improve order picking performance and to support managerial decisions on facing current challenges in order picking (e.g., completing a large number of customer orders within tight delivery schedules). In Section 6.1, we summarize the contents and main contributions to research and practice of this dissertation. Section 6.2 provides directions for future research opportunities.

6.1 Summary and conclusion

This section summarizes the contents of the thesis and lists its main contributions.

Conceptual and methodological fundamentals

In Chapter 2, we give a description of the conceptual and methodological fundamentals that are relevant for the problems addressed in this work. We first outline basic warehouse operations with a focus on order picking. To give an overview of the wide range of order picking systems, we use a classification that differentiates between picker-to-parts systems (mainly involving human order pickers in the order picking) and parts-to-picker systems (mainly involving automated machines). We focus on picker-to-parts systems due to their prevalence in the literature and in practice, and therefore, we detail their central components (i.e., the warehouse layout and the

basic order picking process). Next, we present frequently used planning objectives in order picking and the central planning problems in picker-to-parts setups (i.e., warehouse layout design, storage assignment methods, picker routing strategies, order batching, zoning, and AGV-assisted order picking). We review the literature on these problems and conclude the following with respect to the problems (standard OBP, precedence-constrained order picking, and AOPP) addressed in this work:

- *Item storage assignment*: Studies on storage assignment focus on random storage. Item-specific characteristics such as weight, hazardousness, and temperature requirements, are often neglected when assigning items to storage locations. Likewise, the impact of storage assignment on picker routing is hardly investigated.
- *Picker routing*: Our survey of the literature on picker routing shows that the standard SPRP is the most well-studied PRP, for which mostly exact solution methods have been proposed. Although plenty of research deals with variants of the standard SPRP, precedence-constrained order picking is rather unexplored.
- *Order batching*: With respect to the standard OBP, research focuses on developing heuristic and metaheuristic solution methods to solve the problem. The performance of the OBP methods is often assessed on small-sized instances, and practically relevant instances are either not addressed, or the proposed methods lack in solution quality and/or runtime required to solve such instances. Although ALNS is known to provide a convincing performance on combinatorial optimization problems related to the standard OBP, it has not been developed to address the problem so far.
- *Zone picking*: The few research papers on zoning focus on determining the number of zones and on investigating different item storage assignment strategies in zone picking systems.
- *AGV-assisted order picking*: There is currently only one publication that deals with AGV-assisted order picking. However, the synchronization of human order pickers and AGVs has not been studied.

Finally, Chapter 2 details TS and ALNS as the main components of our solution methods.

Order batching

In Chapter 3, we study the standard OBP to optimize the batching of customer orders with the objective of minimizing the total length of order picking tours. We present a mathematical model formulation of the problem and develop a hybrid solution approach that combines the diversification capabilities of an ALNS and the intensification capabilities of a TS method. Our ALNS \times TS uses an SA-based acceptance criterion to further diversify the search.

In numerical studies, we conduct an extensive comparison of ALNS \times TS to all previously published OBP methods that used standard benchmark sets to investigate their performance. ALNS \times TS outperforms all comparison methods with respect to average solution quality and runtime. Compared to the state-of-the-art, ALNS \times TS shows the clearest advantages on the larger instances of the existing benchmark sets, which assume a larger number of customer orders and larger capacities of the picking device. Finally, ALNS \times TS is able to solve newly generated large-scale instances with up to 600 customer orders and six items per customer order with reasonable runtimes and convincing scaling behavior and robustness.

Precedence-constrained order picking

Chapter 4 addresses a problem based on a practical case, which is inspired by a warehouse of a German manufacturer of household products. In this warehouse, heavy items are not allowed to be placed on top of light items during picking to prevent damage to the light items. Currently, the case company determines the sequence for retrieving the items from their storage locations by applying a simple S-shape strategy that neglects this precedence constraint. As a result, order pickers place the collected items next to each other in plastic boxes and sort the items respecting the precedence constraint.

To avoid this sorting at the end of the order picking process, we propose a picker routing strategy that incorporates the precedence constraint by picking heavy items before light items (E-PRSW). To evaluate E-PRSW, we develop an exact solution method. The algorithm determines the optimal tour of minimum travel length of an order picker for collecting heavy items before light items of a given customer order on a single order picking tour.

We assess the performance of E-PRSW on a dataset provided to us by the manufacturer. We compare E-PRSW to the strategy used in the warehouse of the case company, and to an exact picker routing approach that does not consider the given

precedence constraint. The results clearly demonstrate the convincing performance of E-PRSW even if we compare our strategy to the exact solution method that neglects the precedence constraint.

Furthermore, our experiments show that we improve the order picking process of the case company in the following aspects:

- The case company can avoid the use of plastic boxes by placing the retrieved items directly in the cardboard boxes used for shipping the items to the respective customers.
- The immense sorting effort after the retrieval process is avoided by collecting heavy items before light items.
- The average tour length of an order picker for completing customer orders is significantly reduced in comparison to the results obtained with the picker routing strategy applied in the case company.

Last, we examine the influence of different problem parameters on E-PRSW, and we derive managerial insights for dealing with the given precedence constraint in order picking. An interesting finding from practitioner’s perspective is that by separating heavy and light items in the warehouse and assigning heavy items to storage locations that are arranged close to the depot, a strong reduction of the average tour length is achieved.

AGV-assisted order picking

In Chapter 5, we investigate a new order picking problem, in which human order pickers of the traditional picker-to-parts setup are supported by AGVs. We introduce two mathematical model formulations of the problem, and we develop a two-stage heuristic (ALNS/NEH) to solve the NP-hard problem.

In numerical studies, we first examine the effect of the SA-based acceptance criterion used by ALNS/NEH after the ALNS phase instead of an ALNS/NEH heuristic which only accepts improving solutions. The studies show the positive impact of using the SA-based acceptance criterion instead of simply accepting only improving solutions. Next, we assess the solution quality of ALNS/NEH in comparison to CPLEX solutions. The results demonstrate the ability of ALNS/NEH in finding high-quality solutions within a negligible computation time. ALNS/NEH always provides the optimal solution if CPLEX finds an optimum within the given solution time limit. On all instances on which CPLEX is not able to find the optimal

solution within the runtime limit, ALNS/NEH finds a solution equal to the upper bound obtained by CPLEX. Last, we conduct several computational experiments to investigate the effect of different numbers of AGVs and different traveling and walking speed ratios between AGVs and order pickers on the average total tardiness. From the perspective of warehouse managers, the most interesting findings might be the following:

- The results of our experiments indicate that by adding (or removing) AGVs or by increasing (or decreasing) the AGV speed to adapt to different workloads, a large number of customer orders can be completed until the respective due date.
- One of the most interesting findings is that (in most cases) a slight increase in the speed ratio leads to a larger reduction of the average total tardiness compared to an expansion of the AGV fleet.
- Speed ratios of $\sigma \geq 2.0$ only contribute to minor additional reductions of the average total tardiness compared to a speed ratio of $\sigma = 1.5$. This finding might be interesting for warehouse managers: for safety reasons, speed ratios of $\sigma = 1.0$ or $\sigma = 1.5$ are the norm in real-world warehouses.

6.2 Outlook on future research

This section presents future research opportunities related to the problems addressed in this thesis.

Order batching

An interesting topic for future research related to our ALNS \times TS could be granular neighborhoods used within the TS component. By using sparsification methods, only elements that are likely to be part of high quality solutions remain in the neighborhood. Thus, the runtime may be further reduced without compromising solution quality.

From a practical perspective, it is probably desirable to carry out research on dynamic order batching. Here, information about the incoming customer orders (e.g., requested items and demand quantities) is not known at the beginning of the planning period but becomes available over time.

Moreover, precedence constraints could be incorporated into the standard OBP and our solution method. Item-specific characteristics (e.g., fragility and weight) may influence the retrieval sequence, however, they are generally neglected when dealing with the standard OBP so far.

Precedence-constrained order picking

With respect to our precedence-constrained PRP, we assume only two item weight classes. Considering more weight categories could provide further insights into the performance of the proposed picker routing strategy.

To address our precedence-constrained PRP, other picker routing strategies could be studied. For instance, a heuristic picker routing strategy that determines the retrieval sequence when sorting is carried out while picking but is still easy for order pickers to implement.

Moreover, our problem assumes a one-dimensional stacking system when placing items in a cardboard box required for shipping the items to the respective customer. Because this assumption is only applicable to a few cases in practice, a three-dimensional stacking system could be considered by future research.

AGV-assisted order picking

Human order picking in cooperation with AGVs is one of the latest warehousing technologies that is becoming increasingly popular. However, AGV-assistance in traditional picker-to-parts systems has not yet been sufficiently studied. As mentioned earlier, there is currently one publication on AGV-assistance in traditional picker-to-parts systems. Obviously, there is plenty of room for further scientific consideration. For instance, our AOPP could be extended to address other warehouse layouts (e.g., multi-block parallel-aisle warehouse layouts and chevron warehouse layouts).

Another extension could concern the routing of AGVs: allowing AGVs to arbitrarily travel along the cross aisle instead of a prescribed route (as assumed in this dissertation) could further increase order picking efficiency.

Existing item storage assignment strategies such as turnover-based storage and complementarity-based storage aim at storing frequently requested items in storage locations that are arranged close to the depot. However, in our AGV-assisted picker-to-parts system, also the workload of order pickers, which may vary with different

items storage assignment strategies, could be taken into account. Thus, adapting existing or developing new item storage assignment strategies for our AOPP is another interesting field for future research.

Bibliography

- R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1–3):75–102, 2002.
- M. Albareda-Sambola, A. Alonso-Ayuso, E. Molina, and C. S. de Blas. Variable neighborhood search for order batching in a warehouse. *Asia-Pacific Journal of Operational Research*, 26(5):655–683, 2009.
- K. Azadeh, R. B. M. de Koster, and D. Roy. Robotized and automated warehouse systems: review and recent developments. *Transportation Science*, 53(4):917–945, 2019.
- R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6(2):126–140, 1994.
- C. Blum and A. Roli. Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.
- N. Boysen, R. B. M. de Koster, and F. Weidinger. Warehousing in the e-commerce era: a survey. *European Journal of Operational Research*, 277(2):396–411, 2019a.
- N. Boysen, K. Stephan, and F. Weidinger. Manual order consolidation with put walls: the batched order bin sequencing problem. *EURO Journal on Transportation and Logistics*, 8(2):169–193, 2019b.
- Y. A. Bozer and J. W. Kile. Order batching in walk-and-pick order picking systems. *International Journal of Production Research*, 46(7):1887–1909, 2008.
- O. Briant, H. Cambazard, D. Cattaruzza, N. Catusse, A.-L. Ladier, and M. Ogier. An efficient and general approach for the joint order batching and picker routing problem. *European Journal of Operational Research*, 2020. DOI: 10.1016/j.ejor.2020.01.059.
- F. Caron, G. Marchet, and A. Perego. Routing policies and COI-based storage policies in picker-to-part systems. *International Journal of Production Research*, 36(3):713–732, 1998.
- F. Caron, G. Marchet, and A. Perego. Optimal layout in low-level picker-to-part systems. *International Journal of Production Research*, 38(1):101–117, 2000.
- M. Çelik and H. Süral. Order picking under random and turnover-based storage policies in fishbone aisle warehouses. *IIE Transactions*, 46(3):283–300, 2014.

- M. Çelik and H. Süral. Order picking in parallel-aisle warehouses with multiple blocks: complexity and a graph theory-based heuristic. *International Journal of Production Research*, 57(3):888–906, 2019.
- T. Chabot, R. Lahyani, L. C. Coelho, and J. Renaud. Order picking problems under weight, fragility and category constraints. *International Journal of Production Research*, 55(21):6361–6379, 2017.
- C. Chackelson, A. Errasti, D. Ciprés, and F. Lahoz. Evaluating order picking performance trade-offs by configuring main operating strategies in a retail distributor: a design of experiments approach. *International Journal of Production Research*, 51(20):6097–6109, 2013.
- M.-C. Chen and H.-P. Wu. An association-based clustering approach to order batching considering customer demand patterns. *Omega*, 33(4):333–343, 2005.
- T.-L. Chen, C.-Y. Cheng, Y.-Y. Chen, and L.-K. Chan. An efficient hybrid algorithm for integrated order batching, sequencing and routing problem. *International Journal of Production Economics*, 159:158–167, 2015.
- C.-Y. Cheng, Y.-Y. Chen, T.-L. Chen, and J. J.-W. Yoo. Using a hybrid approach based on the particle swarm optimization and ant colony optimization to solve a joint order batching and picker routing problem. *International Journal of Production Economics*, 170(Part C):805–814, 2015.
- L. Chirici and K.-S. Wang. Tackling the storage problem through genetic algorithms. *Advances in Manufacturing*, 2(3):203–211, 2014.
- K. Choe and G. Sharp. Small parts order picking: design and operation, 1991. Available online at <https://www2.isye.gatech.edu/~mgoetsch/cali/Logistics%20Tutorial/order/article.htm>, last accessed on March 1, 2020.
- J. J. Coyle, E. J. Bardi, and C. J. Langley. *The Management of Business Logistics. A Supply Chain Perspective*. South-Western/Thomson Learning, 7th edition, 2002.
- F. Dallari, G. Marchet, and M. Melacini. Design of order picking system. *The International Journal of Advanced Manufacturing Technology*, 42(1–2):1–12, 2009.
- R. L. Daniels, J. L. Rummel, and R. Schantz. A model for warehouse order picking. *European Journal of Operational Research*, 105(1):1–17, 1998.
- R. B. M. de Koster. Past and future. Perspectives on material handling, 2015. Available online at <https://repub.eur.nl/pub/79094>, last accessed on March 1, 2020.
- R. B. M. de Koster. Automated and robotic warehouses: developments and research opportunities. *Logistics and Transport*, 38(2):33–40, 2018.
- R. B. M. de Koster and E. S. van der Poort. Routing order pickers in a warehouse: a

- comparison between optimal and heuristic solutions. *IIE Transactions*, 30(5):469–480, 1998.
- R. B. M. de Koster, E. S. van der Poort, and M. Wolters. Efficient order batching methods in warehouses. *International Journal of Production Research*, 37(7):1479–1504, 1999.
- R. B. M. de Koster, T. Le-Duc, and K. J. Roodbergen. Design and control of warehouse order picking: a literature review. *European Journal of Operational Research*, 182(2):481–501, 2007.
- R. B. M. de Koster, T. Le-Duc, and N. Zaerpour. Determining the number of zones in a pick-and-sort order picking system. *International Journal of Production Research*, 50(3):757–771, 2012.
- R. de Santis, R. Montanari, G. Vignali, and E. Bottani. An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses. *European Journal of Operational Research*, 267(1):120–137, 2018.
- R. Dekker, R. B. M. de Koster, K. J. Roodbergen, and H. van Kalleveen. Improving order picking response time at Ankor’s warehouse. *INFORMS Journal on Applied Analytics*, 34(4):303–313, 2004.
- D. D. Eisenstein. Analysis and optimal design of discrete order picking technologies along a line. *Naval Research Logistics*, 55(4):350–362, 2008.
- R. M. Elbert, T. Franzke, C. H. Glock, and E. H. Grosse. The effects of human behavior on the efficiency of routing policies in order picking: the case of route deviations. *Computers & Industrial Engineering*, 111:537–551, 2017.
- E. A. Elsayed. Algorithms for optimal material handling in automatic warehousing systems. *International Journal of Production Research*, 19(5):525–535, 1981.
- E. A. Elsayed and R. G. Stern. Computerized algorithms for order processing in automated warehousing systems. *International Journal of Production Research*, 21(4):579–586, 1983.
- E. A. Elsayed and O. I. Unal. Order batching algorithms and travel time estimation for automated storage/retrieval systems. *International Journal of Production Research*, 27(7):1097–1114, 1989.
- N. Gademann and S. van de Velde. Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, 37(1):63–75, 2005.
- M. Gendreau and J.-Y. Potvin. Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 140(1):189–213, 2005.
- M. Gendreau and J.-Y. Potvin. Tabu search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of metaheuristics*, volume 272 of *International Series in Operations Research & Management Science*, pages 37–55. Springer, 2019.

- D. R. Gibson and G. P. Sharp. Order batching procedures. *European Journal of Operational Research*, 58(1):57–67, 1992.
- C. H. Glock and E. H. Grosse. Storage policies and order picking strategies in U-shaped order picking systems with a movable base. *International Journal of Production Research*, 50(16):4344–4357, 2012.
- C. H. Glock, E. H. Grosse, H. Abedinnia, and S. Emde. An integrated model to improve ergonomic and economic performance in order picking by rotating pallets. *European Journal of Operational Research*, 273(2):516–534, 2019.
- F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.
- F. Glover. Tabu search – part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.
- D. Goeke and M. Schneider. Modeling single picker routing problems in classical and modern warehouses. *Working paper, DPO-2018-11 (version 1, 04.11.2018), Deutsche Post Chair – Optimization of Distribution Networks, RWTH Aachen University*, 2018.
- M. Goetschalckx and J. Ashayeri. Classification and design of order picking. *Logistics World*, 2(2):99–106, 1989.
- M. Goetschalckx and H. D. Ratliff. Order picking in an aisle. *IIE Transactions*, 20(1):53–62, 1988.
- A. E. Gray, U. S. Karmarkar, and A. Seidmann. Design and operation of an order consolidation warehouse: models and application. *European Journal of Operational Research*, 58(1):14–36, 1992.
- E. H. Grosse, C. H. Glock, and R. Ballester-Ripoll. A simulated annealing approach for the joint order batching and order picker routing problem with weight restrictions. *International Journal of Operations and Quantitative Management*, 20(2):65–83, 2014.
- E. H. Grosse, C. H. Glock, M. Y. Jaber, and W. P. Neumann. Incorporating human factors in order picking planning models: framework and research opportunities. *International Journal of Production Research*, 53(3):695–717, 2015.
- J. Gu, M. Goetschalckx, and L. F. McGinnis. Research on warehouse operation: a comprehensive review. *European Journal of Operational Research*, 177(1):1–21, 2007.
- J. Gu, M. Goetschalckx, and L. F. McGinnis. Research on warehouse design and performance evaluation: a comprehensive review. *European Journal of Operational Research*, 203(3):539–549, 2010.
- T. Gudehus. *Logistik 1. Grundlagen, Verfahren und Strategien*. Springer, 2012.
- K. R. Gue and R. D. Meller. Aisle configurations for unit-load warehouses. *IIE Transactions*, 41(3):171–182, 2009.

- S. T. Hackman, M. J. Rosenblatt, and J. M. Olin. Allocating items to an automated storage and retrieval system. *IIE Transactions*, 22(1):7–14, 1990.
- R. W. Hall. Distance approximations for routing manual pickers in a warehouse. *IIE Transactions*, 25(4):76–87, 1993.
- P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- W. H. Hausman, L. B. Schwarz, and S. C. Graves. Optimal storage assignment in automatic warehousing systems. *Management Science*, 22(6):629–638, 1976.
- S. Henn. Algorithms for on-line order batching in an order picking warehouse. *Computers & Operations Research*, 39(11):2549–2563, 2012.
- S. Henn. Order batching and sequencing for the minimization of the total tardiness in picker-to-part warehouses. *Flexible Services and Manufacturing Journal*, 27(1):86–114, 2015.
- S. Henn and V. Schmid. Metaheuristics for order batching and sequencing in manual order picking systems. *Computers & Industrial Engineering*, 66(2):338–351, 2013.
- S. Henn and G. Wäscher. Tabu search heuristics for the order batching problem in manual order picking systems. *European Journal of Operational Research*, 222(3):484–494, 2012.
- S. Henn, S. Koch, K. F. Doerner, C. Strauss, and G. Wäscher. Metaheuristics for the order batching problem in manual order picking systems. *Business Research*, 3(1):82–105, 2010.
- S. Henn, S. Koch, and G. Wäscher. Order batching in order picking warehouses: a survey of solution approaches. In R. Manzini, editor, *Warehousing in the global supply chain: advanced models, tools and applications for storage systems*, pages 105–137. Springer, 2012.
- J. L. Heskett. Cube-per-order index – a key to warehouse stock location. *Transportation and Distribution Management*, 3(1):27–31, 1963.
- Y.-C. Ho and S.-P. Chien. A comparison of two zone-visitation sequencing strategies in a distribution centre. *Computers & Industrial Engineering*, 50(4):426–439, 2006.
- Y.-C. Ho, T.-S. Su, and Z.-B. Shi. Order batching methods for an order picking warehouse with two cross aisles. *Computers & Industrial Engineering*, 55(2):321–347, 2008.
- S. Hong and Y. Kim. A route-selecting order batching model with the S-shape routes in a parallel-aisle order picking system. *European Journal of Operational Research*, 257(1):185–196, 2017.

- S. Hong, A. L. Johnson, and B. A. Peters. Batch picking in narrow-aisle order picking systems with consideration for picker blocking. *European Journal of Operational Research*, 221(3):557–570, 2012.
- H. Hwang and D. G. Kim. Order batching heuristics based on cluster analysis in a low-level picker-to-part warehousing system. *International Journal of Production Research*, 43(17):3657–3670, 2005.
- C.-C. Jane. Storage location assignment in a distribution center. *International Journal of Physical Distribution & Logistics Management*, 30(1):55–71, 2000.
- J. M. Jarvis and E. D. McDowell. Optimal product layout in an order picking warehouse. *IIE Transactions*, 23(1):93–102, 1991.
- C. Kallina and J. Lynn. Application of the cube-per-order index rule for stock location in a distribution warehouse. *INFORMS Journal on Applied Analytics*, 7(1):37–46, 1976.
- S. Koch. *Genetische Algorithmen für das Order Batching-Problem in manuellen Kommissioniersystemen*. Springer Gabler, 2014.
- S. Koch and G. Wäscher. A grouping genetic algorithm for the order batching problem in distribution warehouses. *Journal of Business Economics*, 86(1–2):131–153, 2016.
- O. Kulak, Y. Sahin, and M. E. Taner. Joint order batching and picker routing in single and multiple cross aisle warehouses using cluster-based tabu search algorithms. *Flexible Services and Manufacturing Journal*, 24(1):52–80, 2012.
- T. Le-Duc and R. B. M. de Koster. Travel time estimation and order batching in a 2-block warehouse. *European Journal of Operational Research*, 176(1):374–388, 2007.
- C.-M. Liu. Clustering techniques for stock location and order picking in a distribution center. *Computers & Operations Research*, 26(10–11):989–1002, 1999.
- M. Löffler, N. Boysen, and M. Schneider. Picker routing in AGV-assisted order picking systems. *Working paper, Deutsche Post Chair – Optimization of Distribution Networks, RWTH Aachen University*, 2020.
- C. J. Malmborg. Storage assignment policy tradeoffs. *International Journal of Production Research*, 34(2):363–378, 1996.
- C. J. Malmborg and K. Bhaskaran. On the optimality of the cube per order index for conventional warehouses with dual command cycles. *Material Flow*, 4(3):169–175, 1987.
- M. Masae, C. H. Glock, and E. H. Grosse. Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*, 2019a. DOI: 10.1016/j.ijpe.2019.107564.
- M. Masae, C. H. Glock, and P. Vichitkunakorn. Optimal order picker routing in the chevron warehouse. *IIE Transactions*, 2019b. DOI: 10.1080/24725854.2019.1660833.

- M. Masae, C. H. Glock, and P. Vichitkunakorn. Optimal order picker routing in a conventional warehouse with two blocks and arbitrary starting and ending points of a tour. *International Journal of Production Research*, 2020. DOI: 10.1080/00207543.2020.1724342.
- Material Handling Institute. Automatic guided vehicles, 2020. Available online at <http://www.mhi.org/fundamentals/automatic-guided-vehicles>, last accessed on March 1, 2020.
- M. Matusiak, R. B. M. de Koster, L. Kroon, and J. Saarinen. A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. *European Journal of Operational Research*, 236(3):968–977, 2014.
- M. Matusiak, R. B. M. de Koster, and J. Saarinen. Utilizing individual picker skills to improve order batching in a warehouse. *European Journal of Operational Research*, 263(3):888–899, 2017.
- M. Melacini, S. Perotti, and A. Tumino. Development of a framework for pick-and-pass order picking system design. *The International Journal of Advanced Manufacturing Technology*, 53(9–12):841–854, 2011.
- R. D. Meller and K.-Y. Gau. The facility layout problem: recent and emerging trends and perspectives. *Journal of Manufacturing Systems*, 15(5):351–366, 1996.
- N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.
- İ. Muter and T. Öncan. An exact solution approach for the order batching problem. *IIE Transactions*, 47(7):728–738, 2015.
- M. Napolitano. 2012 warehouse/DC operations survey: mixed signals. *Modern Materials Handling*, 51(11):48–56, 2012.
- M. Nawaz, E. E. Ensore, and I. Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983.
- T. Öncan. MILP formulations and an iterated local search algorithm with tabu thresholding for the order batching problem. *European Journal of Operational Research*, 243(1):142–155, 2015.
- Ö. Öztürkoğlu, K. R. Gue, and R. D. Meller. Optimal unit-load warehouse designs for single-command operations. *IIE Transactions*, 44(6):459–475, 2012.
- C.-H. Pan and S.-Y. Liu. A comparative study of order batching algorithms. *Omega*, 23(6):691–700, 1995.
- J. C.-H. Pan and M.-H. Wu. A study of storage assignment problem for an order picking

- line in a pick-and-pass warehousing system. *Computers & Industrial Engineering*, 57(1):261–268, 2009.
- L. Pansart, N. Catusse, and H. Cambazard. Exact algorithms for the order picking problem. *Computers & Operations Research*, 100:117–127, 2018.
- P. J. Parikh and R. D. Meller. Selecting between batch and zone order picking strategies in a distribution center. *Transportation Research Part E: Logistics and Transportation Review*, 44(5):696–719, 2008.
- P. J. Parikh and R. D. Meller. Estimating picker blocking in wide-aisle order picking systems. *IIE Transactions*, 41(3):232–246, 2009.
- B. C. Park. Order picking: issues, systems and models. In R. Manzini, editor, *Warehousing in the global supply chain: advanced models, tools and applications for storage systems*, pages 1–30. Springer, 2012.
- D. W. Pentico. Assignment problems: a golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793, 2007.
- C. G. Petersen. Routeing and storage policy interaction in order picking operations. *Decision Sciences Institute Proceedings*, 3:1614–1616, 1995.
- C. G. Petersen. An evaluation of order picking routeing policies. *International Journal of Operations & Production Management*, 17(11):1098–1111, 1997.
- C. G. Petersen. The impact of routing and storage policies on warehouse efficiency. *International Journal of Operations & Production Management*, 19(10):1053–1064, 1999.
- C. G. Petersen. Considerations in order picking zone configuration. *International Journal of Operations & Production Management*, 22(7):793–805, 2002.
- C. G. Petersen and G. Aase. A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics*, 92(1):11–19, 2002.
- C. G. Petersen and R. W. Schmenner. An evaluation of routing and volume-based storage policies in an order picking operation. *Decision Sciences*, 30(2):481–501, 1999.
- D. Pisinger and S. Ropke. Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of metaheuristics*, volume 272 of *International Series in Operations Research & Management Science*, pages 99–127. Springer, 2019.
- L. M. Pohl, R. D. Meller, and K. R. Gue. Optimizing fishbone aisles for dual-command operations in a warehouse. *Naval Research Logistics*, 56(5):389–403, 2009.
- H. D. Ratliff and A. S. Rosenthal. Order picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521, 1983.
- K. J. Roodbergen. *Layout and routing methods for warehouses*. PhD thesis, ERIM PhD

- Series Research in Management, Erasmus University Rotterdam, the Netherlands, 2001.
- K. J. Roodbergen and R. B. M. de Koster. Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 133(1):32–43, 2001a.
- K. J. Roodbergen and R. B. M. de Koster. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883, 2001b.
- K. J. Roodbergen and I. F. A. Vis. A model for warehouse layout. *IIE Transactions*, 38(10):799–811, 2006.
- K. J. Roodbergen, G. P. Sharp, and I. F. A. Vis. Designing the layout structure of manual order picking areas in warehouses. *IIE Transactions*, 40(11):1032–1045, 2008.
- S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006a.
- S. Ropke and D. Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775, 2006b.
- M. B. Rosenwein. An application of cluster analysis to the problem of locating items within a warehouse. *IIE Transactions*, 26(1):101–103, 1994.
- B. Rouwenhorst, B. Reuter, V. Stockrahm, G. J. van Houtum, R. J. Mantel, and W. H. M. Zijm. Warehouse design and control: framework and literature review. *European Journal of Operational Research*, 122(3):515–533, 2000.
- A. Scholz and G. Wäscher. Order batching and picker routing in manual order picking systems: the benefits of integrated routing. *Central European Journal of Operations Research*, 25(2):491–520, 2017.
- A. Scholz, S. Henn, M. Stuhlmann, and G. Wäscher. A new mathematical programming formulation for the single-picker routing problem. *European Journal of Operational Research*, 253(1):68–84, 2016.
- A. Scholz, D. Schubert, and G. Wäscher. Order picking with multiple pickers and due dates – simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems. *European Journal of Operational Research*, 263(2):461–478, 2017.
- G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000.
- P. Shaw. A new local search algorithm providing high-quality solutions to vehicle routing problems. Technical report, Department of Computer Science, University of Strathclyde, Glasgow, 1997.

- P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, volume 1520, pages 417–431, 1998.
- K. Sörensen. Metaheuristics – the metaphor exposed. *International Transactions in Operational Research*, 22:3–18, 2015.
- R. L. Speaker. Bulk order picking. *Industrial Engineering*, 7(12):14–18, 1975.
- SSI Schäfer. Fahrerloses Transportsystem 2Pick für Behälter, Kartons, Paletten, Stückgutkommissionierung, 2016. Available online at <https://www.youtube.com/watch?v=1qkT6Nwg9CM>, last accessed on March 1, 2020.
- Statista. Total retail sales worldwide from 2017 to 2023, 2019. Available online at <https://www.statista.com/statistics/443522/global-retail-sales>, last accessed on March 1, 2020.
- G. Strack and Y. Pochet. An integrated model for warehouse and inventory planning. *European Journal of Operational Research*, 204(1):35–50, 2010.
- E.-G. Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons Inc., 2009.
- M. ten Hompel, V. Sadowsky, and M. Beck. *Kommissionierung. Materialflusssysteme 2 – Planung und Berechnung der Kommissionierung in der Logistik*. Springer, 2011.
- C. Theys, O. Bräysy, W. Dullaert, and B. Raa. Using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, 200(3):755–763, 2010.
- J. A. Tompkins, J. A. White, Y. A. Bozer, and J. M. A. Tanchoco. *Facilities Planning*. John Wiley & Sons Inc., 4th edition, 2010.
- C.-Y. Tsai, J. J. H. Liou, and T.-M. Huang. Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time. *International Journal of Production Research*, 46(22):6533–6555, 2008.
- C. A. Valle and J. E. Beasley. Order batching using an approximation for the distance travelled by pickers. *European Journal of Operational Research*, 284(2):460–484, 2020.
- C. A. Valle, J. E. Beasley, and A. S. da Cunha. Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, 262(3):817–834, 2017.
- J. P. van den Berg, G. P. Sharp, A. J. R. M. N. Gademann, and Y. Pochet. Forward-reserve allocation in a warehouse with unit-load replenishments. *European Journal of Operational Research*, 111(1):98–113, 1998.
- J. P. van der Gaast. *Stochastic models for order picking systems*. PhD thesis, ERIM PhD

- Series Research in Management, Erasmus University Rotterdam, the Netherlands, 2016.
- T. van Gils, K. Ramaekers, A. Caris, and R. B. M. de Koster. Designing efficient order picking systems by combining planning problems: state-of-the-art classification and review. *European Journal of Operational Research*, 267(1):1–15, 2018.
- I. van Nieuwenhuyse and R. B. M. de Koster. Evaluating order throughput time in 2-block warehouses with time window batching. *International Journal of Production Economics*, 121(2):654–664, 2009.
- D. L. van Oudheusden, Y.-J. J. Tzen, and H.-T. Ko. Improving storage and order picking in a person-on-board AS/R system: a case study. *Engineering Costs and Production Economics*, 13(4):273–283, 1988.
- T. S. Vaughan. The effect of warehouse cross aisles on order picking efficiency. *International Journal of Production Research*, 37(4):881–897, 1999.
- R. Walter, N. Boysen, and A. Scholl. The discrete forward-reserve problem – allocating space, selecting products, and area sizing in forward order picking. *European Journal of Operational Research*, 229(3):585–594, 2013.
- F. Weidinger. Picker routing in rectangular mixed shelves warehouses. *Computers & Operations Research*, 95:139–150, 2018.
- F. Weidinger and N. Boysen. Scattered storage: how to distribute stock keeping units all around a mixed-shelves warehouse. *Transportation Science*, 52(6):1412–1427, 2018.
- J. Won and S. Olafsson. Joint order batching and order picking in warehouse operations. *International Journal of Production Research*, 43(7):1427–1442, 2005.
- G. Wäscher. Order picking: a survey of planning problems and methods. In H. Dyckhoff, R. Lackes, and J. Reese, editors, *Supply chain management and reverse logistics*, pages 323–347. Springer, 2004.
- X. Xiang, C. Liu, and L. Miao. Storage assignment and order batching problem in Kiva mobile fulfilment system. *Engineering Optimization*, 50(11):1941–1962, 2018.
- M. Yu and R. B. M. de Koster. Performance approximation and design of pick-and-pass order picking systems. *IIE Transactions*, 40(11):1054–1069, 2008.
- M. Yu and R. B. M. de Koster. The impact of order batching and picking area zoning on order picking system performance. *European Journal of Operational Research*, 198(2):480–490, 2009.

Appendix A

Alternative mathematical model for the AGV-assisted order picking problem

In the following, we present an alternative mathematical model formulation (AOPP-BI) for our AOPP. Contrary to the AOPP-BE model presented in Section 5.1, batches are not generated before solving a test instance in the AOPP-BI. For convenience, we reintroduce all necessary notation to make this section self-contained.

Note that M denotes a sufficiently large positive number, which must be at least as large as the maximum completion time over all batches. However, because the completion time of a batch is not known before solving a specific test instance, we calculate M based on the worst case scenario according to which a batching of customer orders is not allowed, and customer orders are sequentially picked, i.e., it is not possible to process several customer orders simultaneously. Then, M can be calculated as follows:

$$M = \sum_{o \in O} \left(\max_{i \in V} \{r_{oi} + w_{oi}\} + \sum_{i \in V} l_{oi} + u_o + 2 \cdot \max_{i, j \in V_{0, n+1}} \{t_{ij}\} \right) \quad (\text{A.1})$$

Parameters r_{oi} denote the time the order picker requires to pick those items of customer order $o \in O$ which are stored in her picking aisle $i \in V$, parameters w_{oi} indicate the time the order picker spends on walking to retrieve the items of customer order $o \in O$ which are stored in her picking aisle $i \in V$, parameters l_{oi} specify the time the order picker requires to pass the items of customer order $o \in O$ at handover

location $i \in V$ to the associated AGV, and parameters u_o represent the time required to unload the items of customer order $o \in O$ from the associated AGV. As introduced above, parameters t_{ij} denote the travel time from depot/handover location $i \in V_0$ to handover location/depot $j \in V_{n+1}$.

Using the notation given in Table A.1, the AOPP-BI can be formulated as a mixed integer problem consisting of objective function (A.2) and constraints (A.3) to (A.22) as follows.

$0, n+1$	depot instances
Sets	
B	set of (initially) empty batches, where $ B = O $ because there are at most $ O $ elements in B if each customer order is defined as a single batch (indices: b, d)
O	set of customer orders (indices: o, o')
V	set of picking aisles, where each handover location $i \in V$ is identified through the associated picking aisle $i \in V$ (indices: i, j)
V_0	set of depot instance 0 and picking aisles, where $V_0 = V \cup \{0\}$ (indices: i, j)
V_{n+1}	set of depot instance $n+1$ and picking aisles, where $V_{n+1} = V \cup \{n+1\}$ (indices: i, j)
$V_{0, n+1}$	set of depot instances and picking aisles, where $V_{0, n+1} = V \cup \{0\} \cup \{n+1\}$ (indices: i, j)
Parameters	
c_o	number of items contained in customer order $o \in O$
C	capacity of an AGV
d_o	due date of customer order $o \in O$
l_{oi}	time the order picker requires to pass the items of customer order $o \in O$ at handover location $i \in V$ to the associated AGV
m	number of AGVs
M	a sufficiently large positive number
r_{oi}	time the order picker requires to retrieve the items of customer order $o \in O$ which are stored in picking aisle $i \in V$
t_{ij}	time an AGV requires to travel from depot/handover location $i \in V_0$ to depot/handover location $j \in V_{n+1}$
u_o	time required to unload the items of customer order $o \in O$ from the associated AGV
w_{oi}	time the order picker spends on walking to retrieve the items of customer order $o \in O$ which are stored in picking aisle $i \in V$
η_{oi}	1, if depot/handover location $i \in V_{0, n+1}$ has to be visited by the AGV handling customer order $o \in O$; 0, otherwise
Continuous decision variables	
a_{bi}	arrival time of the AGV handling batch $b \in B$ at handover location $i \in V$
f_b	completion time of batch $b \in B$
h_{bi}	order picker's start time of passing the items of batch $b \in B$ to the associated AGV at handover location $i \in V$
s_{bi}	order picker's start time of picking the items of batch $b \in B$ at handover location $i \in V$
τ_o	tardiness of customer order $o \in O$
Binary decision variables	
α_b	1, if batch $b \in B$ is the first batch handled by a certain AGV; 0, otherwise
γ_{bi}	1, if the depot/handover location $i \in V_{0, n+1}$ has to be visited by the AGV handling batch $b \in B$; 0, otherwise
v_{bo}	1, if customer order $o \in O$ is included in batch $b \in B$; 0, otherwise
x_{bij}	1, if the AGV handling batch $b \in B$ travels from depot/handover location $i \in V_0$ to depot/handover location $j \in V_{n+1}$; 0, otherwise
z_{bd}	1, if batch $b \in B$ is handled before batch $d \in B$ by the order pickers; 0, otherwise
ζ_{bd}	1, if batch $b \in B$ is handled directly before batch $d \in B$ by a certain AGV; 0, otherwise

Table A.1: Overview of the notation used in the AOPP-BI model.

$$\text{minimize } \sum_{o \in O} \tau_o \quad (\text{A.2})$$

subject to

$$\sum_{b \in B} v_{bo} = 1 \quad \forall o \in O \quad (\text{A.3})$$

$$\sum_{o \in O} v_{bo} \cdot c_o \leq C \quad \forall b \in B \quad (\text{A.4})$$

$$\sum_{o \in O} v_{bo} \cdot \eta_{oi} \geq \gamma_{bi} \quad \forall b \in B; i \in V_{0,n+1} \quad (\text{A.5})$$

$$\sum_{o \in O} v_{bo} \cdot \eta_{oi} \leq M \cdot \gamma_{bi} \quad \forall b \in B; i \in V_{0,n+1} \quad (\text{A.6})$$

$$\sum_{\substack{j \in V_{n+1} \\ j > i}} x_{bij} = \gamma_{bi} \quad \forall b \in B; i \in V_0 \quad (\text{A.7})$$

$$\sum_{\substack{i \in V_0 \\ i < j}} x_{bij} = \gamma_{bj} \quad \forall b \in B; j \in V_{n+1} \quad (\text{A.8})$$

$$x_{b,0,n+1} \leq 0 \quad \forall b \in B \quad (\text{A.9})$$

$$z_{bd} + z_{db} = 1 \quad \forall b, d \in B; b \neq d \quad (\text{A.10})$$

$$\alpha_d + \sum_{\substack{b \in B \\ b \neq d}} \zeta_{bd} \geq \gamma_{d,0} \quad \forall d \in B \quad (\text{A.11})$$

$$\sum_{\substack{d \in B \\ d \neq b}} \zeta_{bd} \leq \gamma_{b,0} \quad \forall b \in B \quad (\text{A.12})$$

$$\sum_{b \in B} \alpha_b \leq m \quad (\text{A.13})$$

$$h_{bi} + \sum_{o \in O} v_{bo} \cdot l_{oi} - M \cdot (1 - z_{bd}) \leq s_{di} \quad \forall b, d \in B; b \neq d; i \in V \quad (\text{A.14})$$

$$s_{bi} + \sum_{o \in O} v_{bo} \cdot r_{oi} + v_{bo'} \cdot w_{o'i} \leq h_{bi} \quad \forall b \in B; o' \in O; i \in V \quad (\text{A.15})$$

$$a_{bi} \leq h_{bi} \quad \forall b \in B; i \in V \quad (\text{A.16})$$

$$h_{bi} + \sum_{o \in O} v_{bo} \cdot l_{oi} + t_{ij} - M \cdot (1 - x_{bij}) \leq a_{bj} \quad \forall b \in B; i \in V_0; j \in V; i \neq j \quad (\text{A.17})$$

$$f_b + t_{0,i} - M \cdot (2 - x_{d,0,i} - \zeta_{bd}) \leq a_{di} \quad \forall b, d \in B; b \neq d; i \in V \quad (\text{A.18})$$

$$h_{bi} + \sum_{o \in O} v_{bo} \cdot (l_{oi} + u_o) + t_{i,n+1} - M \cdot (1 - x_{b,i,n+1}) \leq f_b \quad \forall b \in B; i \in V \quad (\text{A.19})$$

$$f_b - d_o - M \cdot (1 - v_{bo}) \leq \tau_o \quad \forall b \in B; o \in O \quad (\text{A.20})$$

$$a_{bi}, f_b, h_{bi}, s_{bi}, \tau_o \geq 0 \quad \forall b \in B; o \in O; i \in V_{0,n+1} \quad (\text{A.21})$$

$$x_{bij}, z_{bd}, \alpha_b, \gamma_{bi}, \zeta_{bd}, v_{bo} \in \{0, 1\} \quad \forall b, d \in B; o \in O; i, j \in V_{0,n+1} \quad (\text{A.22})$$

The objective of minimizing the total tardiness of all customer orders is defined in

(A.2). Constraints (A.3) guarantee that each customer order is assigned to exactly one batch. The capacity of an AGV is considered by satisfying constraints (A.4). Constraints (A.5) and (A.6) in combination with constraints (A.7) and (A.8) state that an AGV collecting the items of a batch starts from the depot, drives to the relevant handover locations from the leftmost to the rightmost, and then returns to the depot. It is also assured that the relevant handover locations are visited exactly once on each AGV tour. Constraints (A.9) exclude direct AGV trips between the depot instances.

Constraints (A.10) define the sequence according to which the batches are to be processed by the order pickers. The sequence in which the batches are processed by AGVs is modeled in constraints (A.11), (A.12), and (A.13). Constraints (A.11) enforce that each selected batch is either the first batch or direct predecessor batch of another batch (or multiple batches) handled by the same AGV. Constraints (A.12) assure that each selected batch has at most one direct successor batch. Constraints (A.13) state that at most one batch can be the first batch handled by a single AGV.

Constraints (A.14) define the order picker's start time of picking the batch items stored in her picking aisle. If a batch $d \in B$ is not the first batch processed by the order picker, we link the order picker's start time of picking the items of batch $d \in B$ to the time at which the order picker has handed over the items of the direct predecessor batch $b \in B$ to the associated AGV. Constraints (A.15) and (A.16) guarantee that the order picker cannot hand over the picked items to the appropriate AGV until (i) she has returned to the handover location with these items (see constraints (A.15)), and (ii) the AGV has arrived at her handover location (see constraints (A.16)). Constraints (A.17) determine the arrival time of the AGV handling batch $b \in B$ at each handover location $j \in V$. Constraints (A.18) link the arrival time of the AGV handling batch $d \in B$ at handover location $i \in V$ to the completion time of batch $b \in B$ (plus the time the AGV requires to travel from the depot to handover location $i \in V$) if the following holds: (i) batch $b \in B$ is handled by the same AGV as batch $d \in B$, and (ii) batch $b \in B$ is processed before batch $d \in B$ by the AGV. Constraints (A.19) compute the completion time for each batch $b \in B$, and constraints (A.20) calculate the tardiness for each customer order $o \in O$. Finally, the continuous decision variables and the binary decision variables are defined in constraints (A.21) and (A.22), respectively.

Appendix B

Detailed computational results of the comparison methods on small instances of our AGV-assisted order picking problem

In the following, we provide detailed computational results for each of the small-sized instances. Results are aggregated by the number of customer orders (5, 6, and 7) in Tables B.1–B.3. In the first column, we give the instance name. For CPLEX and ALNS/NEH, the table reports the total tardiness of all customer orders (column τ) and the average runtime in seconds (column t (s)). Note that CPLEX is given a solution time limit of 1800 seconds, so optimality of the reported solutions is proven by CPLEX whenever $t < 1800$ holds.

Instance	CPLEX		ALNS/NEH	
	τ	t (s)	τ	t (s)
5orders1agv6capacity-1	70	72.3	70	0.085
5orders1agv6capacity-2	103	52.3	103	0.028
5orders1agv6capacity-3	92	47.0	92	0.011
5orders1agv6capacity-4	220	52.3	220	0.011
5orders1agv6capacity-5	232	43.0	232	0.006
5orders1agv6capacity-6	61	63.4	61	0.082
5orders1agv6capacity-7	226	39.2	226	0.022
5orders1agv6capacity-8	210	118.0	210	0.013
5orders1agv6capacity-9	629	104.3	629	0.008
5orders1agv6capacity-10	112	27.0	112	0.007
5orders1agv8capacity-1	136	130.7	136	0.071
5orders1agv8capacity-2	191	42.1	191	0.018
5orders1agv8capacity-3	53	19.1	53	0.005
5orders1agv8capacity-4	319	66.9	319	0.005
5orders1agv8capacity-5	42	84.6	42	0.006
5orders1agv8capacity-6	41	46.3	41	0.005
5orders1agv8capacity-7	514	47.0	514	0.013
5orders1agv8capacity-8	146	42.6	146	0.020
5orders1agv8capacity-9	297	45.7	297	0.015
5orders1agv8capacity-10	10	43.7	10	0.006
5orders2agvs6capacity-1	237	11.5	237	0.009
5orders2agvs6capacity-2	0	1.7	0	0.006
5orders2agvs6capacity-3	58	15.8	58	0.007
5orders2agvs6capacity-4	216	13.8	216	0.004
5orders2agvs6capacity-5	61	15.6	61	0.006
5orders2agvs6capacity-6	404	23.4	404	0.005
5orders2agvs6capacity-7	137	30.8	137	0.011
5orders2agvs6capacity-8	196	11.1	196	0.014
5orders2agvs6capacity-9	62	14.4	62	0.008
5orders2agvs6capacity-10	0	2.1	0	0.016
5orders2agvs8capacity-1	34	14.7	34	0.020
5orders2agvs8capacity-2	184	12.6	184	0.006
5orders2agvs8capacity-3	15	11.8	15	0.014
5orders2agvs8capacity-4	4	32.9	4	0.005
5orders2agvs8capacity-5	87	10.2	87	0.005
5orders2agvs8capacity-6	51	12.3	51	0.005
5orders2agvs8capacity-7	203	5.7	203	0.018
5orders2agvs8capacity-8	239	24.9	239	0.007
5orders2agvs8capacity-9	313	28.6	313	0.009
5orders2agvs8capacity-10	312	23.1	312	0.007

Table B.1: Comparison of the results obtained with CPLEX and ALNS/NEH on the small-sized instances assuming five customer orders. The table reports the name of the instance (column Instance), the total tardiness of all customer orders (column τ), and the average runtime in seconds (column t (s)). CPLEX is given a solution time limit of 1800 seconds, so optimality is not guaranteed for the results which consumed the full time.

Instance	CPLEX		ALNS/NEH	
	τ	t (s)	τ	t (s)
6orders1agv6capacity-1	497	1800.0	497	0.089
6orders1agv6capacity-2	647	1800.0	647	0.037
6orders1agv6capacity-3	1057	1800.0	1057	0.007
6orders1agv6capacity-4	1170	1800.0	1170	0.008
6orders1agv6capacity-5	607	1800.0	607	0.004
6orders1agv6capacity-6	293	1800.0	293	0.010
6orders1agv6capacity-7	849	1800.0	849	0.011
6orders1agv6capacity-8	845	1800.0	845	0.018
6orders1agv6capacity-9	1123	1800.0	1123	0.015
6orders1agv6capacity-10	794	1800.0	794	0.011
6orders1agv8capacity-1	267	1800.0	267	0.040
6orders1agv8capacity-2	102	1800.0	102	0.007
6orders1agv8capacity-3	403	1800.0	403	0.015
6orders1agv8capacity-4	92	1800.0	92	0.008
6orders1agv8capacity-5	224	1800.0	224	0.015
6orders1agv8capacity-6	168	1800.0	168	0.006
6orders1agv8capacity-7	186	1800.0	186	0.005
6orders1agv8capacity-8	527	1800.0	527	0.008
6orders1agv8capacity-9	319	1800.0	319	0.022
6orders1agv8capacity-10	0	8.8	0	0.017
6orders2agvs6capacity-1	217	949.5	217	0.032
6orders2agvs6capacity-2	4	313.8	4	0.010
6orders2agvs6capacity-3	5	485.6	5	0.007
6orders2agvs6capacity-4	303	1746.8	303	0.009
6orders2agvs6capacity-5	137	1800.0	137	0.007
6orders2agvs6capacity-6	357	1800.0	357	0.013
6orders2agvs6capacity-7	247	1125.2	247	0.013
6orders2agvs6capacity-8	0	4.6	0	0.017
6orders2agvs6capacity-9	0	3.5	0	0.020
6orders2agvs6capacity-10	58	1600.3	58	0.011
6orders2agvs8capacity-1	0	3.6	0	0.089
6orders2agvs8capacity-2	173	540.5	173	0.040
6orders2agvs8capacity-3	152	1800.0	152	0.008
6orders2agvs8capacity-4	53	789.3	53	0.007
6orders2agvs8capacity-5	25	116.4	25	0.027
6orders2agvs8capacity-6	98	384.4	98	0.012
6orders2agvs8capacity-7	292	1800.0	292	0.019
6orders2agvs8capacity-8	304	684.2	304	0.016
6orders2agvs8capacity-9	18	181.8	18	0.007
6orders2agvs8capacity-10	94	395.7	94	0.015

Table B.2: Comparison of the results obtained with CPLEX and ALNS/NEH on the small-sized instances assuming six customer orders. The table reports the name of the instance (column Instance), the total tardiness of all customer orders (column τ), and the average runtime in seconds (column t (s)). CPLEX is given a solution time limit of 1800 seconds, so optimality is not guaranteed for the results which consumed the full time.

Instance	CPLEX		ALNS/NEH	
	τ	t (s)	τ	t (s)
7orderslagv6capacity-1	860	1800.0	860	0.135
7orderslagv6capacity-2	10	1800.0	10	0.028
7orderslagv6capacity-3	222	1800.0	222	0.009
7orderslagv6capacity-4	42	1800.0	42	0.014
7orderslagv6capacity-5	709	1800.0	709	0.023
7orderslagv6capacity-6	200	1800.0	200	0.019
7orderslagv6capacity-7	503	1800.0	503	0.010
7orderslagv6capacity-8	0	2.6	0	0.009
7orderslagv6capacity-9	63	1800.0	63	0.019
7orderslagv6capacity-10	264	1800.0	264	0.025
7orderslagv8capacity-1	633	1800.0	633	0.033
7orderslagv8capacity-2	310	1800.0	310	0.018
7orderslagv8capacity-3	565	1800.0	565	0.012
7orderslagv8capacity-4	37	1800.0	37	0.005
7orderslagv8capacity-5	272	1800.0	272	0.013
7orderslagv8capacity-6	216	1800.0	216	0.009
7orderslagv8capacity-7	536	1800.0	536	0.020
7orderslagv8capacity-8	188	1800.0	188	0.013
7orderslagv8capacity-9	103	1800.0	103	0.010
7orderslagv8capacity-10	94	1800.0	94	0.008
7orders2agvs6capacity-1	597	1800.0	597	0.047
7orders2agvs6capacity-2	304	1800.0	304	0.010
7orders2agvs6capacity-3	920	1800.0	920	0.016
7orders2agvs6capacity-4	147	1800.0	147	0.014
7orders2agvs6capacity-5	0	2.7	0	0.015
7orders2agvs6capacity-6	250	1800.0	250	0.022
7orders2agvs6capacity-7	133	1800.0	133	0.015
7orders2agvs6capacity-8	328	1800.0	328	0.019
7orders2agvs6capacity-9	0	7.3	0	0.022
7orders2agvs6capacity-10	0	2.1	0	0.010
7orders2agvs8capacity-1	95	1800.0	95	0.105
7orders2agvs8capacity-2	456	1800.0	456	0.010
7orders2agvs8capacity-3	281	1800.0	281	0.015
7orders2agvs8capacity-4	285	1800.0	285	0.010
7orders2agvs8capacity-5	76	1800.0	76	0.013
7orders2agvs8capacity-6	65	1800.0	65	0.017
7orders2agvs8capacity-7	879	1800.0	879	0.027
7orders2agvs8capacity-8	176	1800.0	176	0.014
7orders2agvs8capacity-9	658	1800.0	658	0.014
7orders2agvs8capacity-10	170	1800.0	170	0.048

Table B.3: Comparison of the results obtained with CPLEX and ALNS/NEH on the small-sized instances assuming seven customer orders. The table reports the name of the instance (column Instance), the total tardiness of all customer orders (column τ), and the average runtime in seconds (column t (s)). CPLEX is given a solution time limit of 1800 seconds, so optimality is not guaranteed for the results which consumed the full time.